39th International Symposium on Theoretical Aspects of Computer Science

STACS 2022, March 15–18, 2022, Marseille, France
(Virtual Conference)

Edited by

Petra Berenbrink
Benjamin Monmege

LIPICS

SYMPOSIUM
ON THEORETICAL
ASPECTS
OF COMPUTER
SCIENCE

LIPIcs – Vol. 219 – STACS 2022 www.dagstuhl.de/lipics

Editors

Petra Berenbrink
University of Hamburg, Germany
berenbrink@informatik.uni-hamburg.de

Benjamin Monmege
Aix-Marseille University, France
benjamin.monmege@univ-amu.fr

ACM Classification 2012
Mathematics of computing → Combinatorics; Mathematics of computing → Graph theory; Theory of computation → Formal languages and automata theory; Theory of computation → Logic; Theory of computation → Design and analysis of algorithms; Theory of computation → Computational complexity and cryptography; Theory of computation → Models of computation

ISBN 978-3-95977-222-8

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-222-8.

Publication date
March, 2022

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

Digital Object Identifier: 10.4230/LIPIcs.STACS.2022.0

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Invited Talks

Regular Papers

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

Preface

The International Symposium on Theoretical Aspects of Computer Science (STACS) conference series is an internationally leading forum for original research on theoretical aspects of computer science. Typical areas are: algorithms and data structures, including: design of parallel, distributed, approximation, parameterized and randomised algorithms; analysis of algorithms and combinatorics of data structures; computational geometry, cryptography, algorithmic learning theory, algorithmic game theory; automata and formal languages, including: algebraic and categorical methods, coding theory; complexity and computability, including: computational and structural complexity theory, parameterised complexity, randomness in computation; logic in computer science, including: finite model theory, database theory, semantics, specification verification, rewriting and deduction; current challenges, for example: natural computing, quantum computing, mobile and net computing, computational social choice.

STACS is held alternately in France and in Germany. This year's conference (taking place virtually from March 15 to 18 in Marseille) is the 39th in the series. Previous meetings took place in Paris (1984), Saarbrücken (1985), Orsay (1986), Passau (1987), Bordeaux (1988), Paderborn (1989), Rouen (1990), Hamburg (1991), Cachan (1992), Würzburg (1993), Caen (1994), München (1995), Grenoble (1996), Lübeck (1997), Paris (1998), Trier (1999), Lille (2000), Dresden (2001), Antibes (2002), Berlin (2003), Montpellier (2004), Stuttgart (2005), Marseille (2006), Aachen (2007), Bordeaux (2008), Freiburg (2009), Nancy (2010), Dortmund (2011), Paris (2012), Kiel (2013), Lyon (2014), München (2015), Orléans (2016), Hannover (2017), Caen (2018), Berlin (2019), Montpellier (2020), and Saarbrücken (2021).

The interest in STACS has remained at a very high level over the past years. The STACS 2022 call for papers led to 203 submissions with authors from 37 countries. Each paper was assigned to three program committee members who, at their discretion, asked external reviewers for reports. For the eighth time within the STACS conference series, there was also a rebuttal period during which authors could submit remarks to the PC concerning the reviews of their papers. In addition, and for the second time, STACS 2022 employed a lightweight double-blind reviewing process: submissions should not reveal the identity of the authors in any way. However, it was still possible for authors to disseminate their ideas or draft versions of their paper as they normally would, for instance by posting drafts on the web or giving talks on their results. The committee selected 57 papers during a four-week electronic meeting held in November and December 2021. This means an acceptance rate around 28%. As co-chairs of the program committee, we would like to sincerely thank all its members and the 403 external reviewers for their valuable work. In particular, there were intense and interesting discussions inside the PC committee. The very high quality of the submissions made the selection an extremely difficult task.

We would like to express our thanks to the four invited speakers: Marie Albenque (LIX, École Polytechnique, France), Maria-Florina Balcan (Carnegie Mellon University, USA), Amina Doumane (CNRS, LIP, ENS Lyon, France), and Fabian Kuhn (University of Freiburg, Germany). STACS 2020 in Montpellier was one of the last conferences that took place physically before the Covid-19 lockdown happened the next week. STACS 2021 took place as a virtual conference as many other conferences before, with pre-recorded videos, short online presentations and discussions, and online social events. We were hoping that STACS 2022 would be proposed as a hybrid conference, with a major proportion of the participants on-site in Marseille. However, the rise of the Covid-19 Omicron variant of the pandemic made this

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

possibility less and less likely. In January 2022, after asking the opinion of the authors, it appeared that at most 30% of the participants still wanted to participate on-site. With the big proportion of online talks that this implies, the experience of on-site participants would be strongly deteriorated. Thus, to improve the quality of the event for all participants, it was decided to turn STACS 2022 as an online event. We hope that the pandemic situation will evolve in such a way that the next STACS events will be able to have an on-site component.

We thank Michael Wagner from the LIPIcs team for assisting us in the publication process and the final production of the proceedings. These proceedings contain extended abstracts of the accepted contributions and abstracts of the invited talks. The authors retain their rights and make their work available under a Creative Commons license. The proceedings are published electronically by Schloss Dagstuhl – Leibniz-Center for Informatics within their LIPIcs series. Finally we would like to thank Aix-Marseille University, Laboratoire d'Informatique et Systèmes, Institut Archimède, and CNRS for their support.

Marseille, March 2022 Petra Berenbrink and Benjamin Monmege

Conference Organization

Program Committee

Akanksha Agrawal	Indian Institute of Technology Madras, India
Eric Allender	Rutgers University, United States of America
Nathalie Aubrun	Université Paris-Saclay, France
Libor Barto	Charles University, Czech Republic
Petra Berenbrink	University of Hamburg, Germany, co-chair
Hans Bodlaender	Uetrecht University, Netherlands
Marthe Bonamy	Bordeaux University, France
David Eppstein	University of California, Irvine, United States of America
Thomas Erlebach	Durham University, United Kingdom
John Fearnley	University of Liverpool, United Kingdom
Yuval Filmus	Technion, Israel
Eldar Fischer	Technion, Israel
Martin Hoefer	University of Frankfurt, Germany
Clemens Kupke	University of Strathclyde, United Kingdom
Massimo Lauria	Sapienza University of Roma, Italy
Troy Lee	University of Technology Sydney, Australia
Karoliina Lehtinen	Aix-Marseille University, CNRS, France
Nicole Megow	University of Bremen, Germany
Marni Mishna	Simon Fraser University, Canada
Benjamin Monmege	Aix-Marseille University, France, co-chair
Angelo Montanari	University of Udine, Italy
Daniel Panario	Carleton University, Canada
Sylvain Perifel	University of Paris, France
Karin Quaas	University of Leipzig, Germany
Heiko Röglin	University of Bonn, Germany
Christian Scheideler	University of Paderborn, Germany
Melanie Schmidt	University of Bonn, Germany
Jeffrey Shallit	University of Waterloo, Canada
Frank Stephan	National University of Singapore, Singapore
Yann Strozecki	Versailles Saint-Quentin-en-Yvelines University, France
Kristina Vušković	University of Leeds, United Kingdom
Osamu Watanabe	Tokyo Institute of Technology, Japan

Steering Committee

Dietmar Berwanger	LMF, CNRS, Université Paris-Saclay
Olaf Beyersdorff	Friedrich-Schiller-Universität Jena
Arnaud Durand	IMJ, Univ. Paris 7
Henning Fernau	Universität Trier
Arne Meier	Leibniz Universität Hannover
Cyril Nicaud	LIGM, Université Paris-Est
Rolf Niedermeier	Technische Universität Berlin
Natacha Portier	LIP, ENS Lyon
Heiko Röglin	Universität Bonn
Sylvain Schmitz	IRIF, Université de Paris
Thomas Schwentick	Technische Universität Dortmund, co-chair
Ioan Todinca	LIFO, Univ. Orléans, co-chair

Local Organizing Committee (Aix-Marseille University)

Nicolas Baudru
Charles Grellois
Arnaud Labourel
Karoliina Lehtinen
Nathan Lhote
Théodore Lopez
Guillaume Maurras
Benjamin Monmege, chair
Etienne Moutot
Julie Parreaux
Pierre-Alain Reynier
Jean-Marc Talbot

Subreviewers

403 external subreviewers assisted the PC. We apologize to any subreviewers who do not appear in this list (because their review was entered manually into Easychair).

Eddie Aamari
Andreas Abel
Andreas Abels
Faisal Abu-Khzam
Erhard Aichinger
Aws Albarghouthi
Shaull Almagor
Andrew Alseth
Andris Ambainis
Ellie Anastasiadi
Antonios Antoniadis
Simon Apers
Anna Arutyunova
Cigdem Aslay
David Auger
John Augustine
Chen Avin
Guy Avni
Yossi Azar
Xavier Badin de Montjoye
Philippe Balbiani
Lorenzo Balzotti
Evripidis Bampis
Nikhil Bansal
Sebastián Barbieri
Georgios Barmpalias
Saugata Basu
Xiaohui Bei
Aleksandrs Belovs
Omri Ben-Eliezer
Massimo Benerecetti
Christoph Berkholz
Olaf Beyersdorff
Ivona Bezakova
Vishwas Bhargava
Pranav Bisht
Thomas Bläsius
Greg Bodwin
Bart Bogaerts
Andrej Bogdanov
Udi Boker
Beate Bollig
Ilario Bonacina
Édouard Bonnet
Flavia Bonomo
Prosenjit Bose
Nicolas Bousquet
Robert Bredereck
Davide Bresolin
Karl Bringmann
Martin Bullinger
Jaroslaw Byrka
Kristóf Bérczi
Ben Cameron
Florent Capelli
Olivier Carton
Antonio Casares
Luca Castelli Aleardi
Matteo Ceccarello
Sayantan Chakraborty
Sourav Chakraborty
Timothy M. Chan
Prerona Chatterjee
Jianer Chen
Christine Cheng
Victor Chepoi
Joe Cheriyan
Rajesh Chitnis
Man Kwun Chiu
Chi-Ning Chou
Sebastian Cioaba
Gil Cohen
Amin Coja-Oghlan
Arjan Cornelissen
Pierre Coucheney
Sam Coy
Pierluigi Crescenzi
Radu Curticapean
Konrad K. Dabrowski
Samir Datta
Sami Davies
Anuj Dawar
Paloma de Lima
Alessandro De Luca
Michel De Rougemont
Ronald de Wolf
Argyrios Deligkas
Holger Dell
Daniele Dell'Erba
Dario Della Monica
Alessandra Di Pierro
Reinhard Diestel
Yann Disser
Henk Don
Agostino Dovier
Rod Downey
Lukas Drexler
Michael Drmota
Fabien Dufoulon
Loric Duhaze
Adrian Dumitrescu
Arnaud Durand
Sven Dziadek
Christoph Dürr
Ahmed El Alaoui
Matthias Englert
Louis Esperet
Jan Eube
Léo Exibard
Marco Faella
Yuri Faenza
Uli Fahrenberg
Uriel Feige
Moran Feldman
Stephen Fenner
Henning Fernau
Angelo Ferrando
Carsten Fischer
Noah Fleming
Hervé Fournier
Paolo Franciosa
Fabian Frei
Tom Friedetzky
Zachary Friggstad
Vincent Froese
Zoltan Fülöp
Nicola Galesi
Michal Garlik
Pawel Gawrychowski
Luca Geatti
Loukas Georgiadis
Nicola Gigante

András Gilyén
Hugo Gimbert
Claude Gravel
Catherine Greenhill
Carla Groenland
Renan Gross
Giovanna Guaiana
Maël Guiraud
Tom Gur
Waldo Gálvez
Thorsten Götte
Magnús M. Halldórsson
Valentina Harizanov
Matthew Harrison-Trainor
Daniel Hausmann
Meng He
Pavol Hell
Luisa Herrmann
Kristian Hinnenthal
Shuichi Hirahara
Denis Hirschfeldt
Ruben Hoeksma
Piotr Hofman
Alexandros Hollender
Felix Hommelsheim
Chien-Chung Huang
Sophie Huiberts
Rupert Hölzl
Christian Ikenmeyer
Zvonko Iljazović
Neil Immerman
Taisuke Izumi
Eli Jaffe
Lars Jaffke
Maximilian Janke
Bruno Jartoux
Jan Johannsen
Konstanty Junosza-Szaniawski
Dominik Kaaser
Shahin Kamali
Naoyuki Kamiyama
Jarkko Kari
Telikepalli Kavitha
David Kempe
Thomas Kesselheim
Shuji Kijima
Eun Jung Kim
Valerie King
Sándor Kisfaludi-Bak
Aleks Kissinger
Hartmut Klauck
Karen Klein
Max Klimm
Peter Kling
Benoît Kloeckner
Yusuke Kobayashi
Michael Kompatscher
Christian Komusiewicz
Christian Konrad
Peter Kostolányi
Michal Koucky
Laszlo Kozma
Jan Kretinsky
Ravishankar Krishnaswamy
Louwe B. Kuijer
Alexander Kulikov
Joshua Marc Könen
Marvin Künnemann
Sébastien Labbe
Bundit Laekhanukit
Manuel Lafond
Michael Lampis
Sophie Laplante
Stephane Le Roux
Van Bang Le
Thierry Lecroq
Jérémy Ledent
Yin Tat Lee
Engel Lefaucheux
Steffen Lempp
Pascal Lenzner
Amit Levi
Moritz Lichter
David Liedtke
Nutan Limaye
Daniel Lokshtanov
Théodore Lopez
Aleksander Łukasiewicz
Ramanujan M. Sridharan
Joshua Maglione
Frederic Magniez
Meena Mahajan
Frederik Mallmann-Trenn
Guillaume Malod
David Manlove
Bodo Manthey
Alberto Marchetti-Spaccamela
Dániel Marx
Arnaud Mary
Elvira Mayordomo
Giuseppe Mazzuoccolo
Moti Medina
Themistoklis Melissourgos
Rafael Mendes de Oliveira
Pranabendu Misra
Rajat Mittal
Matthias Mnich
Joshua Moerman
Sidhanth Mohanty
Bojan Mohar
Morteza Monemizadeh
Mauricio Morales
Pat Morin
Amer Mouawad
Lucia Moura
Evgeny Mozgunov
Marta Mularczyk
Haiko Muller
Carl Mummert
Ian Munro
Dimitrios Myrisiotis
Tobias Mömke
Torsten Mütze
Mikito Nanashima
Reza Naserasr
Gonzalo Navarro
Abhinav Nellore
Eike Neumann
Andre Nies
Susumu Nishimura
Nicolas Nisse
Murphy Yuezhen Niu
Dirk Nowotka
Pierre Ohlmann
Neil Olver
Catherine Oriat
Yota Otachi
Joel Ouaknine
Andreas Padalkin
Soumyabrata Pal
Gopal Pandurangan
Denis Pankratov
Julie Parreaux
Paweł Parys

Ori Parzanchevski
Matthew Patitz
Ami Paz
Seth Pettie
Krzysztof Pietrzak
Alexandru Popa
Nicola Prezza
Gabriele Puppis
Md Lutfar Rahman
Artur Riazanov
Lars Rohwedder
Mirko Rossi
Salvador Roura
Harald Räcke
Pietro Sala
Saket Saurabh
Jens Schlöter
Sylvain Schmitz
Geraud Senizergues
Kazumasa Shinagawa
Blerina Sinaimeri
Benjamin Smith
Shay Solomon
José A. Soto
Ladislav Stacho
Tamon Stephen
Céline Swennenhuis
Marek Szykuła
Suguru Tamaki
Sébastien Tavenas
Mautor Thierry
Patrick Totzke
Jalaj Upadhyay
Erik Jan van Leeuwen
Wessel van Woerden
Ben Lee Volk
Magnus Wahlström
Justin Ward
Daniel Warner
Andreas Wiese
Reem Yassawi
Wei Zhan
Hang Zhou

Francesco Pasquale
Christophe Paul
Andrzej Pelc
Carla Piazza
Thomas Place
Igor Potapov
Kirk Pruhs
David Purser
Malin Rau
Kilian Risse
Andrei Romashchenko
Benjamin Rossman
Alexander Rubtsov
Mathieu Sablik
Ville Salo
Joe Sawada
Marco Schmalhofer
Gregory Schwartzman
Ronen Shaltiel
Sebastian Siebertz
Makrand Sinha
Anastasia Sofronova
Gaurav Sood
Sophie Spirkl
Rafał Stefański
Arne Storjohann
Tibor Szabo
Amnon Ta-Shma
Li-Yang Tan
Stefano Tessaro
Carsten Thomassen
Chris Umans
Henning Urbat
Johan M. M. Van Rooij
Alexandre Vigny
Ulrike von Luxburg
Matthias Walter
Rachel Ward
Dimitri Watel
Guohua Wu
Matteo Zavatteri
Peng Zhang

Ludovic Patey
Ronnie Pavlov
Jannik Peters
Théo Pierron
Supartha Podder
Amaury Pouly
Marcin Przybyłko
Franck Quessette
Dieter Rautenbach
Robert Robere
Frances Rosamond
Peter Rossmanith
Paweł Rzążewski
Kunihiko Sadakane
Srinivasa Rao Satti
Kevin Schewior
Daniel R. Schmidt
Pranab Sen
Alexander Shen
Bertrand Simon
Friedrich Slivovsky
Roberto Solis-Oba
Jonathan Sorenson
Shashank Srivastava
Eckhard Steffen
Dirk Sudholt
Mario Szegedy
Navid Talebanfard
Qiyi Tang
Johan Thapper
Csaba Toth
Seeun William Umboh
Jan van den Brand
Rob van Stee
Jevgēnijs Vihrovs
Tjark Vredeveld
Stefan Walzer
Julian Wargalla
Julian Werthmann
Yinzhan Xu
Meirav Zehavi
Shengyu Zhang

Local Limit of Random Discrete Surface with (Or Without!) a Statistical Physics Model

Marie Albenque ✉ 🏠
LIX, École Polytechnique, CNRS, Palaiseau, France

Abstract

A planar map is an embedding of a planar graph in the sphere, considered up to deformations. A triangulation is a planar map, where all the faces are triangles.

In 2003, in order to define a model of *generic* planar geometry, Angel and Schramm studied the limit of random triangulations on the sphere, [3]. They proved that this model of random maps converges for the Benjamini-Schramm topology (see [4]), or local topology, towards the now famous Uniform Infinite Planar Triangulation (or UIPT), a probability distribution on infinite triangulations, see Figure 1. Soon after, Angel [2] studied some properties of the UIPT. He established that the volume of the balls the UIPT of radius R scales as R^4. Similar results (but with quite different proofs) were then obtained for quadrangulations by Chassaing and Durhuus and Krikun.

The results cited above deal with models of maps that fall in the same "universality class", identified in the physics literature as the class of "pure 2D quantum gravity": the generating series all admit the same critical exponent and the volume of the balls of the local limits of several of those models of random maps are known to grow as R^4. To capture this universal behaviour, a good framework is to consider scaling limits of random maps in the Gromov Hausdorff topology. Indeed, for a wide variety of models the scaling limit exists and is the so-called Brownian map [6, 7], see Figure 2.

To escape this pure gravity behaviour, physicists have long ago understood that one should "couple gravity with matter", that is, consider models of random maps endowed with a statistical physics model. I will present in particular the case of triangulations decorated by an Ising model. It consists in colouring in black and white the vertices of a triangulation, and consider probability distribution which are now biased by their number of monochromatic edges. In a recent work, in collaboration with Laurent Ménard and Gilles Schaeffer [1], we proved that the local limit of this model also exists.

In this talk, I will present these results and explain the main ideas underlying their proof, which rely in part on some enumerative formulas obtained by Tutte in the 60s [8], or their generalization to coloured triangulations by Bernardi and Bousquet-Mélou [5].

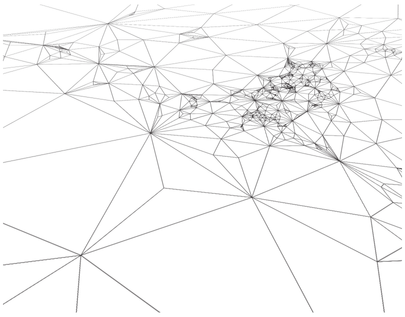


Figure 1 A simulation of a large uniform triangulation, which gives an approximation of the UIPT (© I. Kortchemski).

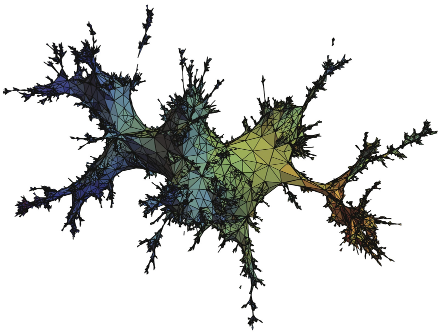


Figure 2 A simulation of a large uniform triangulation, which gives an approximation of the Brownian map (© I. Kortchemski).

2012 ACM Subject Classification Mathematics of computing → Generating functions; Mathematics of computing → Random graphs; Mathematics of computing → Enumeration

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 1; pp. 1:1–1:2
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases Random graphs, triangulations, Benjamini-Schramm convergence, Ising model

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.1

Category Invited Talk

Funding ANR-21-CE48-0007

References

1 M. Albenque, L. Ménard, and G. Schaeffer. Local convergence of large random triangulations coupled with an ising model. *Trans. Amer. Math. Soc.*, 2020.

2 O. Angel. Growth and percolation on the uniform infinite planar triangulation. *Geom. Funct. Anal.*, 13(5):935–974, 2003.

3 O. Angel and O. Schramm. Uniform infinite planar triangulations. *Comm. Math. Phys.*, 241(2-3):191–213, 2003.

4 I. Benjamini and O. Schramm. Recurrence of distributional limits of finite planar graphs. *Electron. J. Probab.*, 6(23), 2001.

5 O. Bernardi and M. Bousquet-Mélou. Counting colored planar maps: algebraicity results. *J. Combin. Theory Ser. B*, 101(5):315–377, 2011.

6 J.-F. Le Gall. Uniqueness and universality of the Brownian map. *Ann. Probab.*, 41(4):2880–2960, 2013.

7 G. Miermont. The Brownian map is the scaling limit of uniform random plane quadrangulations. *Acta Math.*, 210(2):319–401, 2013.

8 W.T. Tutte. A census of planar triangulations. *Canad. J. Math.*, 14:21–38, 1962.

Generalization Guarantees for Data-Driven Mechanism Design

Maria-Florina Balcan
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Many mechanisms including pricing mechanisms and auctions typically come with a variety of tunable parameters which impact significantly their desired performance guarantees. Data-driven mechanism design is a powerful approach for designing mechanisms, where these parameters are tuned via machine learning based on data. In this talk I will discuss how techniques from machine learning theory can be adapted and extended to analyze generalization guarantees of data-driven mechanism design.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Theory of computation → Design and analysis of algorithms

Keywords and phrases mechanism configuration, algorithm configuration, machine learning, generalization guarantees

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.2

Category Invited Talk

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 2; pp. 2:1–2:1
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Deterministic Distributed Symmetry Breaking at the Example of Distributed Graph Coloring

Fabian Kuhn ✉
University of Freiburg, Germany

Abstract

The problem of obtaining fast deterministic algorithms for distributed symmetry breaking problems in graphs has long been considered one of the most challenging problems in the area of distributed graph algorithms. Consider for example the distributed coloring problem, where a (computer) network is modeled by an arbitrary graph $G = (V, E)$ and the objective is to compute a vertex coloring of G by running a distributed algorithm on the graph G. It is maybe not surprising that randomization can be a helpful tool to efficiently compute such a coloring. In fact, as long as each node $v \in V$ can choose among $\deg(v)+1$ different colors, even an almost trivial algorithm in which all nodes keep trying a random available color allows to color all nodes in $O(\log n)$ parallel steps. How to obtain a similarly efficient deterministic distributed coloring algorithm is far less obvious. In fact, for a long time, there has been an exponential gap between the time complexities of the best randomized and the best deterministic distributed algorithms for various graph coloring variants and for many other basic graph problems. In the last few years, there however has been substantial progress on deterministic distributed graph algorithms that are nearly as fast as randomized algorithms for the same tasks. In particular, in a recent breakthrough, Rozhoň and Ghaffari managed to reduce the gap between the randomized and deterministic complexities of locally checkable graph problems to at most $\operatorname{poly}\log n$.

In the talk, we give a brief overview of the history of the problem of finding fast deterministic algorithms for distributed symmetry breaking problems and of what we know about the relation between deterministic and randomized distributed algorithms for such problems. Together with some additional recent developments, the result of Rozhoň and Ghaffari provides a generic, somewhat brute-force way to efficiently derandomize randomized distributed algorithms. Apart from this, there has also been substantial progress on more direct, problem-specific algorithms. In the talk, we in particular discuss some novel deterministic distributed graph coloring algorithms. The algorithms are signficantly faster and we believe also simpler than previous algorithms for the same coloring problems.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases distributed graph algorithms, derandomization, distributed coloring

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.3

Category Invited Talk

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 3; pp. 3:1–3:1
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Mapping Networks via Parallel kth-Hop Traceroute Queries

Ramtin Afshar
University of California-Irvine, CA, USA

Michael T. Goodrich
University of California-Irvine, CA, USA

Pedro Matias
University of California-Irvine, CA, USA

Martha C. Osegueda
University of California-Irvine, CA, USA

Abstract

For a source node, v, and target node, w, the `traceroute` command iteratively issues "kth-hop" queries, for $k = 1, 2, \ldots, \delta(v, w)$, which return the name of the kth vertex on a shortest path from v to w, where $\delta(v, w)$ is the distance between v and w, that is, the number of edges in a shortest-path from v to w. The `traceroute` command is often used for network mapping applications, the study of the connectivity of networks, and it has been studied theoretically with respect to biases it introduces for network mapping when only a subset of nodes in the network can be the source of `traceroute` queries. In this paper, we provide efficient network mapping algorithms, that are based on kth-hop `traceroute` queries. Our results include an algorithm that runs in a constant number of parallel rounds with a subquadratic number of queries under reasonable assumptions about the sampling coverage of the nodes that may issue kth-hop `traceroute` queries. In addition, we introduce a number of new algorithmic techniques, including a high-probability parametric parallelization of a graph clustering technique of Thorup and Zwick, which may be of independent interest.

2012 ACM Subject Classification Theory of computation → Parallel algorithms; Networks → Network algorithms; Mathematics of computing → Graph theory; Mathematics of computing → Probabilistic algorithms

Keywords and phrases Network mapping, graph algorithms, parallel algorithms, distributed computing, query complexity, kth-hop queries

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.4

1 Introduction

Network mapping involves inferring the topology of a communication network, such as the Internet, from queries, e.g., see Figure 1 and [24, 49]. A prominent technique for network mapping is *active probing* using the Unix `traceroute` command to perform queries that reveal routing-path information, e.g., see [24, 32, 49]. We formulate the *network mapping* problem as follows. Suppose we are given access to a subset, $U \subseteq V$, of the vertices of a connected, undirected, unweighted graph, $G = (V, E)$, so that the *distance*, $\delta(u, v)$, between two vertices, u and v, in G is defined as the number of edges on a shortest path joining u and v in G. The n vertices in U are known, but the set of edges, E, is unknown. The subset U represents *vantage point* nodes from which we may issue the following type of queries:

- kth-hop(k, u, v): return the vertex, w, that is the kth vertex on a shortest path from u to v in G. If $k \geq \delta(u, v)$, then return v.

Note that for $u, v \in U$, kth-hop(k, u, v) returns vertices in a single shortest path from u to v. Shortest paths in G are not necessarily unique, however. So, for example, if $\delta(u, v) = \delta(u, w) + \delta(w, v)$, it is not necessarily the case that kth-hop$(\delta(u, w), u, v) = w$. In

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 4; pp. 4:1–4:21
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

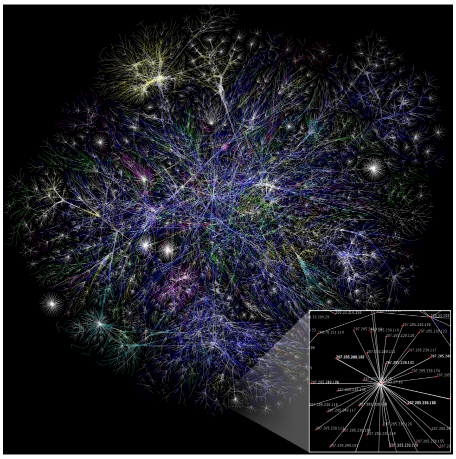


Figure 1 Partial map of the Internet circa 2005. Image by The Opte Project, unchanged and licensed under the Creative Commons Attribution 2.5 Generic license.

the network mapping problem, we are interested in using kth-hop queries to learn the edges of the *induced shortest-path graph*, $H = (U, \tilde{E})$, such that there is an edge $(u, v) \in \tilde{E}$, for $u, v \in U$, if and only if no kth-hop(k, u, v) query would return a vertex $w \in U$ other than v, that is, kth-hop(k, u, v) would return vertices of a shortest path from u to v that does not include any other vertex in U. Thus, H is a weighted, connected, undirected graph such that each edge (u, v) in H has weight $\delta(u, v)$.

Our motivation for focusing on kth-hop queries is that they form the "inner loop" of how `traceroute` works by default. In particular, by default `traceroute` works by sending a series of packets in a network from a source, u, to a destination, v, with the packets having increasing time-to-live (TTL) values, up to an upper bound for the diameter, diam(G), of G, which `traceroute` typically sets to 30 or 64 by default depending on the underlying operating system. The TTL field in a packet is decremented with each hop it traverses and when it reaches 1, then that node sends an ICMP message to the source address (with message including the node's address), e.g., see [1,2]. Thus, the `traceroute` tool can be viewed as first performing a kth-hop$(1, u, v)$ query, then a kth-hop$(2, u, v)$ query, and so on, until getting a response from the vertex v. In fact, one can use options with the `traceroute` command to issue a kth-hop query directly, e.g., to find the 5th hop from a node to `example.com`, one could use the command, "`traceroute -m 5 -M 5 example.com`".

Our formulation of the network mapping problem abstracts away certain system issues. In particular, we are implicitly assuming that messages in G are routed along shortest paths, which is a widely used setting assumed by the prior work [4,20,26]. An important system issue that we do not abstract away, however, is that only vertices in $U \subseteq V$ may issue queries. Indeed, there is some interesting prior work regarding the sampling biases introduced by only being able to issue queries from a subset, U, of the set of vertices, V, in G. For example, Achlioptas, Clauset, Kempe, and Moore [4] show that `traceroute` sampling[1] finds power-law degree distributions in both Δ-regular and Poisson-distributed random graphs, even though these underlying graphs do not themselves have power-law degree distributions, which is a statistical finding in experiments by Lakhina, Byers, Crovella, and Xie [37]. Maciej,

[1] *Traceroute sampling* samples the network graph as the union of paths that packets traverse in performing `traceroute` queries from a subset of the nodes in a network.

Markopoulou, and Patrick [36] study ways to correct for this bias when samping large graphs. Further, Zhang, Kolaczyk, and Spencer [50] and Flaxman and Vera [26] study ways to correct for this bias for estimating degree distributions. Interestingly, Barrat, Alvarez-Hamelin, Dall'Asta, Vázquez, and Vespignani [13] provide an analysis that power laws still exist in the Internet graph in spite of the `traceroute` sampling bias, which these authors show is related to betweenness (see also [20]).

In spite of this interesting prior work concerning the sampling biases inherent in performing `traceroute` queries only from the nodes in the subset, U, we are not familiar with any prior work on efficient algorithms for solving the network mapping problem. We focus on two complexity measures for a network mapping algorithm, $\mathcal{A}$, in terms of $n = |U|$:

- $Q(n)$: the *query complexity* of $\mathcal{A}$. This is the total number of kth-hop queries issued. This complexity measure comes from learning theory (e.g., see [5, 18, 22, 43]) and complexity theory (where it is also known as "decision-tree complexity," e.g., see [16, 48]).
- $R(n)$: the *round complexity* of $\mathcal{A}$. This is the number of rounds of querying performed by $\mathcal{A}$, where the queries issued in a round are given in a batch such that any query issued in a round may not depend on the response to any other query in that round (but each query may depend on results of queries from previous rounds).

Prior Related Work. As mentioned above, we are not aware of prior algorithmic work on network mapping. If we analyze the algorithm used in existing mapping systems that use active probing, this amounts to a brute-force quadratic algorithm implemented by cooperating nodes of the network, which perform a `traceroute` to every other known node in the network, e.g., see [23, 24, 33]. Viewed combinatorially, this algorithm has query complexity, $Q(n)$, that is $O(\text{diam}(G) \cdot n^2)$, and round complexity, $R(n)$, that is $O(\text{diam}(G))$, for kth-hop queries.

The network mapping problem is related to *graph reconstruction*, e.g., see [3, 6, 7, 9–12, 14, 15, 17, 18, 21, 27–30, 34, 35, 38, 40–42, 44, 46, 47]. In this problem, one is given a connected unweighted graph, $G = (V, E)$, for which V is known and goal is to discover E through queries, such as:

- distance(u, v): return the distance, $\delta(u, v)$, between u to v in G.
- shortest-path(u, v): return the vertices (in order) in a shortest path from u to v in G.

There is also work on other types of queries, including vertex-betweenness queries [3]; queries returning whether a given subset of vertices induce a given edge [9–12, 15, 17]; queries returning the number of edges induced by a given subset of vertices [18, 27–29]; queries returning *all* shortest paths from a given node to all other nodes [14, 42]; queries returning the distance between two leaves in a phylogenetic tree [6, 7, 30, 35, 40, 47]; and queries returning whether a given vertex is an ancestor of another given vertex in a rooted tree [6, 7, 46].

There are a number of important differences between the network mapping problem and graph reconstruction, however. Most significantly, the graph reconstruction problem assumes queries can be performed for any vertices in V, whereas in the network mapping problem we may only issue kth-hop queries for nodes in the subset $U \subseteq V$. In addition, even if we restrict the network mapping problem to the case where $U = V$, previous work on graph reconstruction has not considered kth-hop queries, which, as we mentioned above, form the "inner-loop" for how `traceroute` works and are distinct from distance and shortest-path queries. For example, it doesn't seem possible to simulate a kth-hop query with fewer than $\Theta(n)$ distance queries, while a distance query can be simulated with $O(\log \text{diam}(G))$ kth-hop queries via binary search. Also, although it is trivial to simulate a kth-hop query with a single shortest-path query, it takes $\Theta(\text{diam}(G))$ kth-hop queries to simulate a single shortest-path query. Thus, kth-hop queries are strictly weaker than shortest-path queries while being better at capturing the true message complexity of the `traceroute` command.

Another difference between the network mapping problem and graph reconstruction is that previous work on graph reconstruction has mostly focused on how to sequentially reconstruct the graph, G, whereas the network mapping problem is inherently parallel, due to the motivation from mapping real-world networks, where each node is a computer. In terms of previous work on graph reconstruction in parallel, Mathieu and Zhou [38] recently provided a simple algorithm to reconstruct a connected, unweighted graph G, using an expected number of $\tilde{O}(N^{5/3})$ distance queries in 2 rounds.[2] They also show that their algorithm takes an expected number of $\tilde{O}(N)$ distance queries to reconstruct a random Δ-regular graphs.

The most relevant prior work on graph reconstruction, however, is by Kannan, Mathieu, and Zhou [34], who show how to reconstruct a connected, unweighted graph, G, using an expected number of $O(\Delta^3 N^{3/2} \log^2 N \log\log N)$ distance queries, or an expected number of $N^{1+O(\tau(N))}$ shortest-path queries, where $N = |V|$ and $\tau(N) = \sqrt{(\log\log N + \log\Delta)/\log N}$, which is $o(1)$ when Δ, the maximum degree of G, is $N^{o(1)}$. They also show that verifying a given set of edges can be done using $O(N^{1+O(\tau(N))})$ expected distance queries.

Our Results. A preliminary announcement of some of this paper's results, using distance queries for graph reconstruction, where queries can be performed for any vertices in V, was presented in [8].

In Section 2, we introduce a new technique that may be of independent interest, where we provide a new parallel implementation of a well-known graph clustering technique of Thorup and Zwick [45] with round complexity of $O(1)$, while their original implementation implies an expected round complexity of $O(\log n)$. In doing so, we introduce a parameter that allows to trade off parallel time and cluster size. Moreover, we show that our complexity bounds hold with high probability,[3] whereas Thorup and Zwick proved their complexity bounds only in expectation. In Section 3, we will use this new construction to compute a graph-theoretic Voronoi diagram in our network mapping algorithm. On the other hand, our graph clustering technique can be applied to other problems, such as that studied by Honiden, Houle, and Sommer [31] for balancing graph-theoretic Voronoi diagrams, to reduce the number of centers to $O(s)$ from $O(s \log n)$.

In Section 3, we provide the first non-trivial algorithmic results for the network mapping problem. Our query complexities and round complexities are characterized in terms of $n = |U|$ and some interesting parameters that capture the sampling coverage provided by the set U. For example, in addition to characterizing complexities in terms of Δ, the maximum degree of the graph, H, we introduce a distance coverage parameter, $\delta_{\max}$, which is the maximum weight for an edge in H, and a nearby-vertices parameter, μ, which is an upper bound on the number of vertices within a distance of $2\delta_{\max}$ of any given vertex $v \in U$. As we show, these parameters are required for the sake of efficiency, for we show that without these parameters the network mapping problem has a quadratic query-complexity lower bound. For example, under reasonable assumptions regarding these parameters, we are the first to give a constant-round network-mapping algorithm with query complexity better than the trivial brute-force algorithm.

In Section 4, we introduce a greedy approach for network mapping that is based on parallel greedy approximate set cover, which allows us to achieve a near-quasilinear query complexity (when Δ is $n^{o(1)}$). As with a related sequential greedy graph reconstruction result of Kannan, Mathieu, and Zhou [34], our query and round complexity bounds are parameterized in terms

[2] The notation $\tilde{O}(f(N))$ is equivalent with $O(f(N) \cdot polylog(f(N)))$.
[3] We say an event holds *with high probability* (w.h.p.) if it occurs with probability at least $1 - 1/n$.

of the best sequential query complexity for verifying the edges of a graph using distance queries (without knowing the exact value of this query complexity). Further, for small values of the parameters, $\delta_{\max}$ and Δ, our greedy approach uses a near-quasilinear number of kth-hop queries, which are strictly weaker than the shortest-path queries used by Kannan, Mathieu, and Zhou. We summarize our results in Table 1.

Table 1 Our w.h.p. bounds for the network mapping problem, where ϵ denotes a fixed constant, $0 < \epsilon < 1/2$, $n = |U|$ and Δ, $\delta_{\max}$, μ, and $\tau(\cdot)$ are as defined above.

$R(n)$	$Q(n)$
$O(1)$	$O(\delta_{\max}\, \mu\, n^{3/2+\epsilon})$
$O(\log n \cdot \log \operatorname{diam}(G))$	$O(\mu\, n^{3/2} \log^{3/2} n \cdot \log \operatorname{diam}(G))$
if $U \subset V$: $O(\Delta n)$	$\operatorname{diam}(G) \cdot n^{1+O(\tau(n))}$
if $U = V$: $O(\Delta n \log n)$	$n^{1+O(\tau(n))}$

2 Parallel Graph Clustering

Thorup and Zwick [45] introduced a graph clustering technique in presenting a stretch[4] 3 network routing scheme. We begin by describing our parallel graph clustering algorithm, which may be of independent interest, as it provides a parameterized parallel extension of the one by Thorup and Zwick [45]. Also, whereas Thorup and Zwick establish their bounds in expectation, we establish ours with high probability. In Section 3, we apply our parallel graph clustering algorithm in creating a graph-theoretic Voronoi diagram for our network mapping algorithm.

We begin with some review from Thorup and Zwick [45]. Let $G = (V, E)$ be a connected, undirected n-vertex graph, and let $\delta(u, v)$ denote the distance between vertices u and v in G. In this section, we allow G to be weighted, where $\delta(u, v)$ is the sum of weights on a shortest path (lowest weight path) from u to v, but in our algorithms for parallel network mapping, we assume G is unweighted, in which case $\delta(u, v)$ is the number of edges on a shortest path from u to v. For a subset $A \subseteq V$, let $\delta(A, v) = \min_{a\in A} \delta(a, v)$, and, for vertices $w, v \in V$, let $C_A(w)$ be the *cluster* of w and $B_A(v)$ be the *bunch* of v with respect to A, defined as follows:

$$C_A(w) = \{v \in V \mid \delta(w, v) < \delta(A, v)\} \quad \text{and} \quad B_A(v) = \{w \in V \mid \delta(w, v) < \delta(A, v)\}.$$

Note that if $w \in A$, then $C_A(w) = \emptyset$. Also, observe that bunches and clusters are "inverses" of each other, in that $v \in C_A(w)$ if and only if $w \in B_A(v)$. In addition, notice that clusters and bunches can only shrink as we add vertices to A; that is, if $A' \subseteq A$, then $C_A(w) \subseteq C_{A'}(w)$ and $B_A(v) \subseteq B_{A'}(v)$, for all v and w in V.

Now, let $\beta \in [4, n)$, be a "parallelism" parameter and let $s \in [4 \ln n, n)$ be a "size" parameter. Define a subset, $A \subseteq V$, to be a set of (β, s)*-balanced centers* if $|C_A(w)| \le \beta n/s$, for all $w \in V$. Thorup and Zwick [45] give a sequential algorithm for finding a set of $(4, s)$-balanced centers of expected size $O(s \log n)$. In Algorithm 1, we give a parallel algorithm for finding a set of (β, s)-balanced centers of size $O(s \log_\beta n)$ in $O(\log_\beta n)$ rounds w.h.p. Thus, the parameter β allows one to trade off parallel time and cluster size.

[4] *Routing Stretch* is the worst ratio between the length of a path on which a message is routed and the length of the shortest path in the network from the source to the destination.

Algorithm 1 parallel-centers(V, s, β).

```
$A \leftarrow \emptyset$, $W \leftarrow V$
while $|W| > 0$ do
    $A' \leftarrow \mathsf{Sample}(W, s)$    // a random sample of expected size $s$ (or $W$ if $s \geq |W|$)
    $A \leftarrow A \cup A'$
    for $w \in W$ do in parallel
        $C_A(w) \leftarrow \{v \in V : \delta(w, v) < \delta(A, v)\}$
    $W \leftarrow \{w \in W : |C_A(w)| > \beta n/s\}$
return $A$
```

Our algorithm (Algorithm 1) takes a graph $G = (V, E)$ as input and initializes A, the eventual output of the algorithm, to be empty, and W, the set of nodes $v \in V$ where $|C_A(v)| > \beta n/s$, to be V. Then, we iteratively add $\mathsf{Sample}(W, s)$ to A, and replace W with vertices $w \in W$ such that $|C_A(w)| > \beta n/s$, in parallel, where the function, $\mathsf{Sample}(W, s)$, returns W if $|W| \leq s$ and, otherwise, returns a set of elements from W such that each element in W is selected independently at random with probability $s/|W|$. We continue in this way until $W = \emptyset$.

Since the size of a cluster, $|C_A(w)|$, does not increase as we add more vertices to A, the set A returned by our algorithm is a set of (β, s)-balanced centers. Also, the Sample function returns a sample of size at most $2s$ with probability at least $1 - e^{-s/3}$, which holds with high probability across all iterations when $s \geq 4 \ln n$, by a standard Chernoff bound, e.g., see [39, p. 69]. Incidentally, Thorup and Zwick use the same Sample function, but don't bound its maximum size as we do. This high-probability upper bound for the sample size is not sufficient to achieve a high-probability bound, however, for the entire parallel graph clustering algorithm.

To that end, we define a parameter, α, as follows:

$$\alpha = \begin{cases} 2 & \text{if } \beta \leq ((4/3)e)^4 \\ (4/3)e\beta^{1/2} & \text{otherwise} \end{cases}$$

where $e \approx 2.71828$ is Euler's number. This definition of α is made so that we may achieve high probability bounds for a range of β values.

Let W_i denote the set W at the beginning of iteration i, let A'_i denote the set A' that was added in iteration i, and let A_i denote the set A in this iteration, including the set, A'_i, i.e., $A_i = A_{i-1} \cup A'_i$, for $i = 1, 2, \ldots$, where $A_0 = \emptyset$. Say that iteration i is "bad" if the following inequality holds:

$$\sum_{w \in W_i} |C_{A'_i}(w)| > \frac{\alpha n |W_i|}{s},$$

and that otherwise it is "good". Note that, since W_i is a given for iteration i, whether iteration i is good or bad depends only on A'_i.

▶ **Lemma 1** (Thorup-Zwick [45], Lemma 3.2). *Let $W \subseteq V$, let $1 \leq s \leq n$, and let $A' \leftarrow$ Sample(W, s). Then, for every $v \in V$, $E[\,|B_{A'}(v) \cap W|\,] \leq |W|/s$.*

This implies the following:

$$E\left[\sum_{w \in W_i} |C_{A'_i}(w)|\right] = E\left[\sum_{v \in V} |B_{A'_i}(v) \cap W_i|\right] \leq \frac{n|W_i|}{s}.$$

Therefore, by Markov's inequality, an iteration is bad with probability at most $1/\alpha$.

Let W_{i+1} denote the set of vertices, W, whose clusters have size at least $\beta n/s$ at the end of a good iteration i. As $W_{i+1} \subseteq W_i$, and $C_{A_i}(w) \subseteq C_{A'_i}(w)$, for all $w \in V$, in a good iteration we have:

$$\frac{\beta n|W_{i+1}|}{s} \le \sum_{w\in W_i} |C_{A_i}(w)| \le \sum_{w\in W_i} |C_{A'_i}(w)| \le \frac{\alpha n|W_i|}{s};$$

hence, $|W_{i+1}| \le (\alpha/\beta)|W_i|$ in a good iteration. Thus, the number of good iterations of our algorithm is $O(\log_{(\beta/\alpha)} n)$, which is $O(\log_\beta n)$ for either choice of α. Moreover, because an iteration is good independent of whether any other iteration is good or bad, we may use standard and non-standard Chernoff bounds to show that the number of bad iterations is also $O(\log_\beta n)$ w.h.p., for either value of α. (See Appendix A.1.) This gives us the following:

▶ **Theorem 2.** *Given an undirected, connected graph, $G = (V, E)$, we can find a set, A, of (β, s)-balanced centers of size $O(s \log_\beta n)$ in $O(\log_\beta n)$ parallel rounds w.h.p.*

For example, if $\beta = 4$, then A is constructed to have size $O(s \log n)$ in $O(\log n)$ rounds; if $\beta = n^\epsilon$, for constant $0 < \epsilon < 1/2$, then A is constructed to have size $O(s)$ in $O(1)$ rounds.

3 Our Fast Parallel Network Mapping Algorithms

In this section, we provide our fast parallel network mapping algorithms for a connected, undirected, unweighted network, $G = (V, E)$, given a subset $U \subseteq V$ from which we may perform kth-hop queries. We denote the size of U by n and the size of V by N. Let H be the graph induced by the shortest paths in G between pairs of vertices in U. That is, H has vertex set U and there is an edge (u, v) in H, for $u, v \in U$, if the shortest path between u and v in G determined by kth-hop queries contains no other vertex in U besides u and v. The weight of each edge (u, v) in H is the distance, $\delta(u, v)$, between u and v in G. The goal of network mapping is to determine the edges of H (which can then be used to easily determine the vertices in G in a shortest path corresponding to each edge (u, v) in H in a single round of $\delta(u, v)$ kth-hop queries). We assume we know the value of $\delta_{\max}$, which is the weight of a maximum-weight edge in H. For example, if $U = V$, then $\delta_{\max} = 1$. In the worst case, $\delta_{\max}$ is equal to the diameter of G, but in real-world network mapping applications, $\delta_{\max}$ is likely to be a constant.

We perform all the queries needed in our parallel network mapping algorithms in a subroutine, Distances(v, W), which determines the distance, $\delta(v, w)$, for a given $v \in U$ and every other $w \in W \subseteq U$. We describe two possible implementations for Distances(v, W), which we choose between depending on our desired goals. In our first implementation, we perform a simple binary search using kth-hop(k, v, w) queries to determine $\delta(v, w)$, for each w. This requires $O(\log \operatorname{diam}(G))$ rounds and a total of $O(|W| \log \operatorname{diam}(G))$ kth-hop queries, and this implementation doesn't require any assumptions about W. Note that we assume that we know $\operatorname{diam}(G)$, and if this is not the case, we can instead perform a doubling binary search with the same query complexity. In our second implementation, we perform $\delta_{\max}$ kth-hop(k, w, v) queries in parallel, for each w, for $k = 1$ to $\delta_{\max}$. This implementation requires a single round of $O(\delta_{\max} |W|)$ kth-hop queries, and it requires that $v \in W$, and the nodes in W induce a connected subgraph of H that contains the shortest path in H from each w in W to v, and that we are only interested in finding the edges of this subgraph. This set of queries finds all the edges of a breadth-first search (BFS) tree, B_v, rooted at v, in the induced graph, H, since a shortest path from w to v is also a shortest path from v

Algorithm 2 Our parallel querying algorithm, estimated-parallel-centers(U, s, β), for finding a set of (β, s)-balanced centers A.

```
A ← ∅, W ← U
$T \leftarrow c_1(s/\beta)\log n$   // $c_1$ is a constant set in the analysis
while $|W| > 0$ do
    $A' \leftarrow$ Sample$(W, s)$
    $A \leftarrow A \cup A'$
    $R \leftarrow$ a random subset with $v \in U$ chosen independently with probability $T/n$
    for each $r \in R$ do in parallel
        Distances$(r, U)$
    for $w \in W$ do in parallel
        $S(w) \leftarrow \{v \in R : \delta(w, v) < \delta(A, v)\}$   // $S(w) = C_A(w) \cap R$
    $W \leftarrow \{w \in W : |S(w)| > 2\beta T/s\}$   // that is, $|S(w)|(n/T) > 2\beta n/s$
return $A$
```

to w, and a subpath of any shortest path is a shortest path for its endpoints. Thus, in this second implementation, we can determine $\delta(v, w)$, for each $w \in U$, from B_v, by summing the weights of the edges from v to w in B_v (which doesn't require any additional queries). This gives us the following lemma.

▶ **Lemma 3.** *Distances(v, W) can be implemented in $O(\log \text{diam}(G))$ rounds using a total of $O(|W| \log \text{diam}(G))$ kth-hop queries. Alternatively, if $v \in W$, and the subgraph of H induced by W is connected and we are interested only in finding the edges of this subgraph, then Distances(v, W) can be implemented in 1 round with $O(\delta_{\max} |W|)$ kth-hop queries.*

Let the cluster of vertex w with respect to centers A be $C_A(w) = \{v \in U \mid \delta(w, v) < \delta(A, v)\}$. The key idea of our parallel network mapping algorithm is to first find a set, A, of (β, s)-balanced centers, using our parallel algorithm from the previous section, and then use this set of centers to compute a graph-theoretic Voronoi diagram [25, 31] for G, from which we may efficiently then perform a brute-force querying step for each Voronoi region. This approach is similar in spirit to the one by Kannan, Mathieu, Zhou [34, Section 2], with some key important differences: i) the restriction of our queries to the vantage point $U \subseteq V$ and the parameters capturing sampling coverage of set U, ii) the usage of kth-hop queries, and iii) our parallel graph clustering that allows us to trade off between round complexity and query complexity.

The initial center-finding step builds a set, A, of size $O(s \log_\beta n)$ such that each vertex in U has a cluster with respect to A of size at most $\beta n/s$. One of the challenges in implementing this algorithm efficiently in parallel using kth-hop queries is that we need to determine cluster sizes for all vertices in U in each iteration, which would take too many queries to compute exactly. So, rather than compute such sizes exactly, we instead build a global random set, R, which we use to approximate the size of each cluster. We give the details in Algorithm 2.

▶ **Lemma 4.** *Our estimated-parallel-centers algorithm constructs a set, A, of $(3\beta, s)$-balanced centers of size $O(s \log_\beta n)$. Suppose Distances(r, U) executes in $R(n)$ rounds and $Q(n)$ kth-hop queries. Then estimated-parallel-centers algorithm executes in $O(R(n) \log_\beta n)$ rounds and $O(Q(n)(s/\beta) \log n \log_\beta n)$ kth-hop queries, w.h.p.*

Proof. See Appendix A.2. ◀

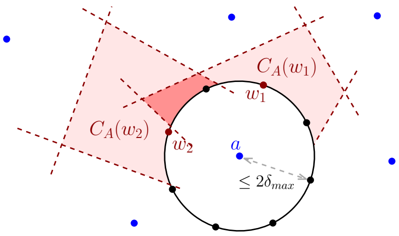


Figure 2 This figure represents a partial structure of our Voronoi Diagram. Blue vertices represent centers from A. The circle centered at $a \in A$ represents the vertices of distance at most $2\delta_{\max}$ from a. We use clusters of nearby vertices of a to discover boundary edges. For simplicity, we draw only two clusters for two arbitrary nodes $w_1, w_2 \in N_{2\delta_{\max}}(a)$.

Now that we have defined and analyzed the function estimated-parallel-centers(U, s, β), let us next turn to our parallel algorithm for mapping a connected, undirected graph, $G = (V, E)$, given a subset, $U \subseteq V$, from which we can perform kth-hop queries. This algorithm takes as input the vertex set U, and outputs, $\tilde{E}$, the set of edges of the induced graph, H, defined by the vertex set U and the shortest paths in G returned by kth-hop queries.

Let $A \subseteq U$ be a set of *centers*, which in our network mapping algorithm will come from a call to our estimated-parallel-centers(U, s, β) algorithm, but a graph-theoretic Voronoi diagram can be defined for any weighted graph and any set of centers. Given a center, $a \in A$, define the *Voronoi cell*, $\mathrm{Vor}_A(a)$, for a in H as $\mathrm{Vor}_A(a) = \{v \in U : \delta(a, v) \leq \delta(A \backslash \{a\}, v)\}$. The *graph-theoretic Voronoi diagram* for A in U consists of the union of Voronoi cells, $\mathrm{Vor}_A(a)$, for each center, $a \in A$. We say that an edge $(v, w) \in \tilde{E}$ is an *interior* edge if $v, w \in \mathrm{Vor}_A(a)$, for some center $a \in A$, and it is a *boundary* edge if $v \in \mathrm{Vor}_A(a)$ and $w \in \mathrm{Vor}_A(b)$, where $a \neq b$. If we were to perform a set of kth-hop queries for every pair of vertices in a Voronoi cell, then we are guaranteed to discover every internal edge in $\mathrm{Vor}_A(a)$, but we will miss boundary edges going between two Voronoi cells. Thus, we need to "branch out" a little bit from the vertices of $\mathrm{Vor}_A(a)$ in order to discover all the boundary edges. To facilitate this, for any center, $a \in A$, let $N_{2\delta_{\max}}(a)$ be the set of "nearby" vertices in H, that is, vertices that are within a distance of $2\delta_{\max}$ of a. Formally,

$$N_{2\delta_{\max}}(a) = \{v \in U : \delta(a, v) \leq 2\delta_{\max}\}.$$

We assume we know μ, the maximum size of $N_{2\delta_{\max}}(a)$, for any $a \in A$. Of course, $\mu < n$. The following lemma shows that it is sufficient to consider these nearby neighbors, for each center $a \in A$, in order to cover all the edges in H, including interior edges and boundary edges. (See also Figure 2.)

We give the details of our network mapping algorithm in Algorithm 3. Through a call to estimated-parallel-centers(U, s, β), we find a set of $(O(\beta), s)$-balanced centers, A. Next, we build a BFS tree from each vertex $a \in A$ to be able to identify nodes in $N_{2\delta_{\max}}(a)$. Then, our mapping algorithm, map, constructs graph-theoretic Voronoi diagram for the centers in A, and then "branches out" from each center $a \in A$ by considering the nodes in $N_{2\delta_{\max}}(a)$ and the clusters defined by nodes in $N_{2\delta_{\max}}(a)$. Finally, after having done this Voronoi decomposition, our algorithm performs exhaustive searches in each cluster in parallel. This part of the algorithm uses a method, Exhaustive-Query(W), which finds all the edges of H between vertices in W by calling Distances(v, W), for each $v \in W$.

The following lemmas establish the correctness and performance complexities for our network mapping algorithm.

Algorithm 3 Parallel network mapping using kth-hop queries.

```
1  Function map($U$):
2      $A \leftarrow$ estimated-parallel-centers($U, s, \beta$)
3      for each $a \in A$ do in parallel
4          Distances($a, U$)      // gives us $N_{2\delta_{\max}}(a)$ as well
5      for each $a \in A$ do in parallel
6          $E_a \leftarrow$ Exhaustive-Query($N_{2\delta_{\max}}(a)$)
7          for $b \in N_{2\delta_{\max}}(a)$ do in parallel
8              Distances($b, U$)
9              $C_A(b) \leftarrow \{v \in U : \delta(b,v) < \delta(A,v)\}$
10             $E_{a,b} \leftarrow$ Exhaustive-Query($C_A(b)$)
11     return $\bigcup_{a \in A} \left( E_a \cup \bigcup_{b \in N_{2\delta_{\max}}(a)} E_{a,b} \right)$
```

▶ **Lemma 5.** *Let (u, v) be an edge in H. Then there exists a center, $a \in A$, such that u and v are both in $N_{2\delta_{\max}}(a)$ or both in $C_A(b)$, for some $b \in N_{2\delta_{\max}}(a)$.*

Proof. Let (u, v) be an edge in H, and note that, by definition, $\delta(u, v) \le \delta_{\max}$. Assume, without loss of generality, that $\delta(A, u) \le \delta(A, v)$. Also, let a be a vertex in A such that $\delta(a, u) = \delta(A, u)$. If $\delta(a, u) \le \delta_{\max}$, then both u and v are in $N_{2\delta_{\max}}(a)$, by the triangle inequality. So, suppose $\delta(a, u) > \delta_{\max}$. Let b be a vertex in U on a shortest path from a to u such that $\delta_{\max} < \delta(a, b) \le 2\delta_{\max}$. Note that b must exist, since no edge in H has weight more than $\delta_{\max}$ (it is possible that $b = u$). Further, b is in $N_{2\delta_{\max}}(a)$ and $\delta(a, u) = \delta(a, b) + \delta(b, u)$. Also, $\delta(b, u) < \delta(a, u) = \delta(A, u)$; hence, u is in $C_A(b)$.

By the triangle inequality, and the above observations,

$$\begin{aligned}
\delta(b, v) &\le \delta(b, u) + \delta(u, v) \\
&\le \delta(b, u) + \delta_{\max} \\
&= \delta(a, u) - \delta(a, b) + \delta_{\max} \\
&< \delta(a, u) - \delta_{\max} + \delta_{\max} \\
&= \delta(a, u) \\
&= \delta(A, u).
\end{aligned}$$

Therefore, $\delta(b, v) < \delta(A, v)$; hence, v is also in $C_A(b)$. ◀

▶ **Lemma 6.** *If Distances(v, W) executes in $R(|W|)$ rounds using $Q(|W|)$ kth-hop queries, then Algorithm 3 uses $O(Q(n)(s/\beta) \log n \log_\beta n + \mu(Q(n) + (\beta n/s)Q(\beta n/s))s \log_\beta n)$ queries in $O(R(n) \log_\beta n)$ rounds, w.h.p., where $\mu = \max_{a \in A} |N_{2\delta_{\max}}(a)|$.*

Proof. By Lemma 4, estimated-parallel-centers(U, s, β) executes in $O(R(n) \log_\beta n)$ rounds and $O(Q(n)(s/\beta) \log n \log_\beta n)$ kth-hop queries, and returns a set of $(3\beta, s)$-balanced centers of size $O(s \log_\beta n)$, w.h.p. The parallel Distances calls in line 4 thus executes in $O(R(n))$ rounds using $O(Q(n)s \log_\beta n)$ kth-hop queries, w.h.p., and the calls to Exhaustive-Query in line 6 execute in $O(R(\mu))$ rounds using a total of $O(\mu Q(\mu)s \log_\beta n)$ kth-hop queries, but these bounds are dominated by the Distances calls in line 8, all of which execute in $O(R(n))$ rounds using $O(\mu Q(n)s \log_\beta n)$ kth-hop queries, w.h.p. Finally, the calls to Exhaustive-Query in line 10 execute in $O(R(\beta n/s))$ rounds using $O(\mu(\beta n/s)Q(\beta n/s)s \log_\beta n)$ kth-hop queries, w.h.p. ◀

Plugging in our derived bounds for Distances, we get the following theorem.

▶ **Theorem 7.** *Given a connected graph, $G = (V, E)$, and subset, $U \subseteq V$, one can map the n-vertex induced shortest-path graph, H, with respect to G and U in $O(1)$ rounds using $O(\delta_{\max}\, \mu\, n^{3/2+\epsilon})$ kth-hop queries, for constant $0 < \epsilon < 1/2$, w.h.p. Alternatively, one can map H in $O(\log n \cdot \log \operatorname{diam}(G))$ rounds using $O(\mu\, n^{3/2} \log^{3/2} n \cdot \log \operatorname{diam}(G))$ queries, w.h.p.*

Proof. For the first result, set $\beta = n^\epsilon$ and $s = n^{1/2+\epsilon}$, and let us use the implementation of Distances that executes in 1 round using $O(\delta_{\max}\, n)$ kth-hop queries from Lemma 3. For the second result, set $\beta = 4$ and $s = (n/\log n)^{1/2}$ and let us use the implementation of Distances that executes in $O(\log \operatorname{diam}(G))$ rounds using $O(n \log \operatorname{diam}(G))$ kth-hop queries from Lemma 3. The bounds follow by Lemma 6. ◀

For example, depending on the values of $\delta_{\max}$ and μ, the above theorem establishes an improvement over the brute-force querying algorithm for solving the parallel network mapping problem in $O(1)$ rounds. The following theorem shows that bounding the parameters $\delta_{\max}$ and μ is needed in order to do better than a quadratic number of kth-hop queries.

▶ **Theorem 8.** *There is an infinite family of n-vertex graphs, G, such that mapping the induced shortest-path graph, H, for a set, U, of $O(n)$ vertices requires $\Omega(n^2)$ kth-hop queries, when μ is $\Theta(n)$ and $\delta_{\max}$ is $\Theta(\log n)$, even if G has maximum degree 3.*

Proof. Let G be the graph of a complete binary tree with n nodes and let U be the set of leaves of G. Thus, the distance in H between any node, v, in the left subtree of G, and a node, w, in the right subtree of G, is $2 \log n$. Thus, $\delta_{\max} = 2 \log n$ and μ is $\Theta(n)$. Now, let G' be G plus a single path in G of length $(2 \log n) - 1$ between two vertices, v and w, in U such that v is in the left subtree of G and w is in the right subtree of G. Thus, there is an edge with weight $(2 \log n) - 1$ joining v and w in the induced shortest-path graph, H', for G', and otherwise H' has the same edge set as H. But there are $\Omega(n^2)$ such possible pairs and the only way to discover the edge (v, w) in H is to perform a kth-hop(k, v, w) query. Any other type of kth-hop query cannot distinguish between H and H'. ◀

4 A Greedy Network Mapping Algorithm

Kannan, Methieu, and Zhou [34] introduce a proof technique that sequentially uses a verification algorithm for unweighted graphs as an oracle for issuing shortest-path queries in a greedy graph reconstruction algorithm. In this section, we show how to adapt this proof technique to a parallel setting and apply it to map the weighted graph, H. For example, our algorithm uses kth-hop queries and provides parallelism according to a parameter, $1 \le p < n$.

Our greedy algorithm is based on performing steps of the classic greedy set cover algorithm in parallel batches. Recall that in this problem, one is given a collection of sets, $S_1, S_2, \ldots, S_m$, whose union is the universe $\mathcal{U}$, and the goal is to find a smallest sub-collection of sets whose union is $\mathcal{U}$, that is, a sub-collection that covers $\mathcal{U}$. The greedy algorithm repeatedly chooses the set covering the maximum number of uncovered items in $\mathcal{U}$, and this results in a number of sets that is at most an $O(\log n)$ factor more than optimum [19].

Let $f(n, \Delta)$ be the query complexity of the best sequential algorithm for the problem of *graph verification* for any connected unweighted graph of n vertices and maximum degree Δ via distance queries, that is, for determining whether an unknown graph, $G = (V, E)$, is equal to a given graph, $\hat{G} = (V, \hat{E})$. For example, Kannan, Methieu, and Zhou [34] show that $f(n, \Delta)$ is $n^{1+O(\tau(n))}$, where $\tau(n) = \sqrt{(\log \log n + \log \Delta)/\log n}$. The function, $f(n, \Delta)$, is used only in our analysis, where we show that, given a parallelism parameter, $1 \le p < n$, our parallel network mapping algorithm can be tuned to have the desired query complexity, $Q(n)$, and round complexity, $R(n)$.

The distance queries in an unweighted graph verification algorithm perform two functions – confirming that edges in $\hat{E}$ are actually in E and confirming that edges not in $\hat{E}$ are also not in E, that is, confirming every $(u,v) \notin \hat{E}$ is not in E. To that latter end, let $\hat{\delta}_h(u,v)$ denote the *hop-count* (number of edges) distance between u and v based on the edges in $\hat{E}$, and let $\hat{E}^c$ denote the set of non-edges in $\hat{G}$, that is, the set of pairs, (u,v), such that $u \neq v$ and $(u,v) \notin \hat{E}$. Similarly, let E^c denote the set of non-edges in G. For any set of tentative edges, $\hat{E}$, define the following set for each pair of vertices, $(u,v) \in \hat{E}^c$: $S_{u,v}(\hat{E}) = \left\{(x,y) \in \hat{E}^c \mid \hat{\delta}_h(u,x) + \hat{\delta}_h(y,v) + 1 < \hat{\delta}_h(u,v)\right\}$.

Kannan, Methieu, and Zhou [34] prove the following two lemmas.

▶ **Lemma 9.** *Suppose $\hat{E} \subseteq E$. For any $(u,v) \in \hat{E}^c$, if $\delta_h(u,v) = \hat{\delta}_h(u,v)$, where $\delta_h(u,v)$ denotes the hop-count distance between u and v in G, then $S_{u,v}(\hat{E}) \subseteq E^c$, that is, each pair in $S_{u,v}(\hat{E})$ is a non-edge in G.*

▶ **Lemma 10.** *If a set of distance queries, T, verifies that every non-edge of $\hat{G}$ is a non-edge of G, then $\cup_{(u,v)\in T} S_{u,v}(\hat{E}) = \hat{E}^c$.*

We present our parallel greedy algorithm for mapping $H = (U, \tilde{E})$ in $G = (V,E)$, for when $U \subset V$, that is, we incrementally build our tentative edge set $\hat{E} \subseteq \tilde{E}$:

1. We initialize a set of tentative edges, $\hat{E}$, to a spanning tree of H by calling kth-hop(k,v,u) from every vertex, $v \in U$, to an arbitrarily chosen vertex, $u \in U$, for $k = 1,\ldots,\mathrm{diam}(G)$, in parallel. We initialize a set of confirmed non-edges of H, $F \leftarrow \emptyset$. Note that we always maintain that $F \subseteq \hat{E}^c$. This requires 1 round of $O(\mathrm{diam}(G)n)$ kth-hop queries.
2. We compute all the $S_{u,v}(\hat{E})$ sets, for pairs $(u,v) \in \hat{E}^c$, which requires no queries.
3. We perform p steps of the greedy set-cover algorithm applied to the sets, $S_{u,v}(\hat{E})\backslash F$, with the goal of covering the remaining pairs, in $\hat{E}^c\backslash F$, in a greedy fashion, which also requires no queries. Let $\{(u_1,v_1),(u_2,v_2),\ldots,(u_p,v_p)\}$ denote the vertex pairs for the $S_{u,v}(\hat{E})$ sets chosen by these greedy steps.
4. We perform kth-hop(k,u_i,v_i) queries, for $k = 1,\ldots,\mathrm{diam}(G)$, in parallel, to determine the actual hop-count distance, $\delta_h(u_i,v_i)$, between u_i and v_i in H, for each $i = 1,2,\ldots,p$ in parallel. This step requires $O(1)$ rounds of $O(p \cdot \mathrm{diam}(G))$ kth-hop queries in total.
5. For each i such that $\delta_h(u_i,v_i) = \hat{\delta}_h(u_i,v_i)$, we add all the pairs in $S_{u_i,v_i}(\hat{E})$ to F. If $F = \hat{E}^c$, then we are done, by Lemma 10.
6. Otherwise, if $F \neq \hat{E}^c$ and $\delta_h(u_i,v_i) = \hat{\delta}_h(u_i,v_i)$, for all $i = 1,2,\ldots,p$, then we repeat the above process, performing another p steps of greedy set cover, looping back to Step 3.
7. If, on the other hand, $F \neq \hat{E}^c$ and $\delta_h(u_i,v_i) < \hat{\delta}_h(u_i,v_i)$, for some i, then there must be at least one edge on a shortest path from u_i to v_i that is in $\tilde{E}$ and not yet in $\hat{E}$. In this case, we add all such edges (which were discovered when we performed the diam(G) kth-hop(k,u_i,v_i) queries) to $\hat{E}$, and repeat the above greedy searching for this updated set, $\hat{E}$, of candidate edges, returning to Step 2.

This gives us the following result.

▶ **Theorem 11.** *Let $f(n,\Delta)$ be the query complexity of an optimal sequential algorithm for graph verification for any unweighted connected graph with n vertices and maximum degree Δ using distance queries. Then, for $1 \leq p < n$, our parallel network mapping algorithm has kth-hop query complexity, $Q(n) \in O((\Delta np + f(n,\Delta)\log n)\mathrm{diam}(G))$ and round complexity, $R(n) \in O((\Delta n + (f(n,\Delta)/p)\log n))$, if $U \subset V$, or $Q(n) \in O((\Delta np + f(n,\Delta)\log n)\log n)$ and round complexity, $R(n) \in O((\Delta n + (f(n,\Delta)/p)\log n)\log n)$, if $U = V$.*

Proof. See Appendix A.3. ◀

Thus, setting p to be $n^{O(\tau(n))}$ gives us the following.

▶ **Corollary 12.** *One can solve the network mapping problem with query complexity, $Q(n)$, that is* $\mathrm{diam}(G) \cdot n^{1+O(\tau(n))}$ *and round complexity, $R(n)$, that is $O(\Delta n)$, if $U \subset V$, or with $Q(n)$ that is $n^{1+O(\tau(n))}$ and round complexity, $R(n)$, that is $O(\Delta n \log n)$, if $U = V$.*

This query complexity is within an $n^{o(1)}$ factor of optimal when Δ is $n^{o(1)}$, by the following simple lower bound.

▶ **Theorem 13.** *Solving the network mapping problem for an n-vertex graph, G, with maximum degree, Δ, requires $\Omega(\Delta n)$ kth-hop queries, even if H has only n edges.*

Proof. Let H be a caterpillar (i.e., a tree where every leaf is at distance 1 from a vertex on a central path), such that every internal node has degree Δ. Choose any pair, u and v, of sibling leaves and connect them with an edge. The only way to discover the edge, (u, v), is to perform a kth-hop(k, u, v) query, for $k \geq 1$. Thus, in expectation, any graph reconstruction algorithm must perform a query for over half of the pairs of siblings in H, that is, at least $\Omega((n/\Delta)\Delta^2) = \Omega(\Delta n)$ queries, in order to discover all the edges of H. ◀

5 Conclusion

We have given efficient algorithms for solving the network mapping problem in parallel. Such algorithms show the effectiveness of kth-hop queries, even though they are weaker than shortest-path queries. Our methods assume knowledge of $\delta_{\max}$ and μ, but this assumption can be relaxed at the expense of increasing the round complexity by an $O(\log n)$ factor, while keeping the query complexity unchanged, by using our algorithm as a blackbox to perform a doubling search for the values of these parameters. Our methods also assume kth-hop(k, u, v) remains same in the algorithm, which is a reasonable assumption in static routing. In our network mapping formulation, we abstracted away some system issues such that when the TTL field of a packet reaches 1, the node sends an ICMP message to the source address; however, in the real Internet, some nodes may have their ICMP responses switched off. Therefore, a direction to extend this work would be to design algorithms addressing such system issues.

We have also given new, parallel implementations for graph clustering, which provide tradeoffs between the number of center vertices and the sizes of clusters. Even for sequential algorithms, this result may prove useful for applications where minimizing the number of center points is a primary optimization goal. For instance, one can apply our construction to the problems studied by Honiden *et al.* [31] for balancing graph-theoretic Voronoi diagrams to shave a $O(\log n)$ factor of the number of centers. It seems likely, therefore, that this result will have other applications as well.

References

1 TRACEROUTE for Linux. http://traceroute.sourceforge.net/. Accessed: April 6, 2020.

2 *TRACEROUTE(8) Traceroute For Linux*, October 2006. Accessed: April 6, 2020. URL: http://man7.org/linux/man-pages/man8/traceroute.8.html.

3 Mikkel Abrahamsen, Greg Bodwin, Eva Rotenberg, and Morten Stöckel. Graph reconstruction with a betweenness oracle. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 5:1–5:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.5.

4 Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 694–703, 2005. doi:10.1145/1060590.1060693.

5 Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, and Kurt Mehlhorn. The query complexity of finding a hidden permutation. In Andrej Brodnik, Alejandro López-Ortiz, Venkatesh Raman, and Alfredo Viola, editors, *Space-Efficient Data Structures, Streams, and Algorithms: Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, pages 1–11, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40273-9_1.

6 Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. Reconstructing binary trees in parallel. In Christian Scheideler and Michael Spear, editors, *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, July 15-17, 2020*, pages 491–492. ACM, 2020. doi:10.1145/3350755.3400229.

7 Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. Reconstructing biological and digital phylogenetic trees in parallel. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 3:1–3:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.3.

8 Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. Parallel network mapping algorithms. In Kunal Agrawal and Yossi Azar, editors, *SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021*, pages 410–413. ACM, 2021. doi:10.1145/3409964.3461822.

9 Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM J. Discrete Math.*, 18(4):697–712, 2005. doi:10.1137/S0895480103431071.

10 Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM J. Comput.*, 33(2):487–501, 2004. doi:10.1137/S0097539702420139.

11 Dana Angluin and Jiang Chen. Learning a hidden hypergraph. *J. Mach. Learn. Res.*, 7:2215–2236, 2006. URL: http://jmlr.org/papers/v7/angluin06a.html.

12 Dana Angluin and Jiang Chen. Learning a hidden graph using O(log n) queries per edge. *J. Comput. Syst. Sci.*, 74(4):546–556, 2008. doi:10.1016/j.jcss.2007.06.006.

13 Alain Barrat, Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alexei Vázquez, and Alessandro Vespignani. Sampling of networks with traceroute-like probes. *Complexus*, 3(1-3):83–96, 2006.

14 Zuzana Beerliova, Felix Eberhard, Thomas Erlebach, Alexander Hall, Michael Hoffmann, Matús Mihalák, and L. Shankar Ram. Network discovery and verification. *IEEE Journal on Selected Areas in Communications*, 24(12):2168–2181, 2006. doi:10.1109/JSAC.2006.884015.

15 Richard Beigel, Noga Alon, Simon Kasif, Mehmet Serkan Apaydin, and Lance Fortnow. An optimal procedure for gap closing in whole genome shotgun sequencing. In Thomas Lengauer, editor, *Proceedings of the Fifth Annual International Conference on Computational Biology, RECOMB 2001, Montréal, Québec, Canada, April 22-25, 2001*, pages 22–30. ACM, 2001. doi:10.1145/369133.369152.

16 Anna Bernasconi, Carsten Damm, and Igor Shparlinski. Circuit and decision tree complexity of some number theoretic problems. *Information and Computation*, 168(2):113–124, 2001. doi:10.1006/inco.2000.3017.

17 Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In Dieter Kratsch, editor, *Graph-Theoretic Concepts in Computer Science, 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers*, volume 3787 of *Lecture Notes in Computer Science*, pages 16–27. Springer, 2005. doi:10.1007/11604686_2.

18 Sung-Soon Choi and Jeong Han Kim. Optimal query complexity bounds for finding graphs. *Artificial Intelligence*, 174(9):551–569, 2010. doi:10.1016/j.artint.2010.02.003.

19 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

20 Luca Dall'Asta, J. Ignacio Alvarez-Hamelin, Alain Barrat, Alexei Vázquez, and Alessandro Vespignani. Exploring networks with traceroute-like probes: Theory and simulations. *Theor. Comput. Sci.*, 355(1):6–24, 2006. doi:10.1016/j.tcs.2005.12.009.

21 Luca Dall'Asta, Ignacio Alvarez-Hamelin, Alain Barrat, Alexei Vázquez, and Alessandro Vespignani. Exploring networks with traceroute-like probes: Theory and simulations. *Theoretical Computer Science*, 355(1):6–24, 2006. URL: http://www.sciencedirect.com/science/article/pii/S0304397505009126.

22 Shahar Dobzinski and Jan Vondrak. From query complexity to computational complexity. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1107–1116, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214076.

23 Benoit Donnet. Internet topology discovery. In Ernst Biersack, Christian Callegari, and Maja Matijasevic, editors, *Data Traffic Monitoring and Analysis - From Measurement, Classification, and Anomaly Detection to Quality of Experience*, volume 7754 of *LNCS*, pages 44–81. Springer, 2013. doi:10.1007/978-3-642-36784-7_3.

24 Benoit Donnet and Timur Friedman. Internet topology discovery: A survey. *IEEE Communications Surveys and Tutorials*, 9(1-4):56–69, 2007. doi:10.1109/COMST.2007.4444750.

25 Martin Erwig. The graph Voronoi diagram with applications. *Networks*, 36(3):156–163, 2000. doi:10.1002/1097-0037(200010)36:3<156::AID-NET2>3.0.CO;2-L.

26 Abraham D. Flaxman and Juan Vera. Bias reduction in traceroute sampling - towards a more accurate map of the internet. In Anthony Bonato and Fan R. K. Chung, editors, *Algorithms and Models for the Web-Graph, 5th International Workshop, WAW 2007, San Diego, CA, USA, December 11-12, 2007, Proceedings*, volume 4863 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007. doi:10.1007/978-3-540-77004-6_1.

27 Vladimir Grebinski. On the power of additive combinatorial search model. In Wen-Lian Hsu and Ming-Yang Kao, editors, *Computing and Combinatorics, 4th Annual International Conference, COCOON '98, Taipei, Taiwan, R.o.C., August 12-14, 1998, Proceedings*, volume 1449 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 1998. doi:10.1007/3-540-68535-9_23.

28 Vladimir Grebinski and Gregory Kucherov. Reconstructing a hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discret. Appl. Math.*, 88(1-3):147–165, 1998. doi:10.1016/S0166-218X(98)00070-5.

29 Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000. doi:10.1007/s004530010033.

30 Jotun J Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bulletin of mathematical biology*, 51(5):597–603, 1989.

31 S. Honiden, M. E. Houle, and C. Sommer. Balancing graph Voronoi diagrams. In *Sixth International Symposium on Voronoi Diagrams*, pages 183–191, 2009.

32 B. Huffaker, D. Plummer, D. Moore, and K. Claffy. Topology discovery by active probing. In *IEEE Symposium on Applications and the Internet (SAINT)*, pages 90–96, 2002.

33 Bradley Huffaker, Daniel Plummer, David Moore, and KC Claffy. Topology discovery by active probing. In *Proceedings 2002 Symposium on Applications and the Internet (SAINT) Workshops*, pages 90–96. IEEE, 2002.

34 Sampath Kannan, Claire Mathieu, and Hang Zhou. Graph reconstruction and verification. *ACM Trans. Algorithms*, 14(4):40:1–40:30, 2018. doi:10.1145/3199606.

35 Valerie King, Li Zhang, and Yunhong Zhou. On the complexity of distance-based evolutionary tree reconstruction. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 444–453, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644179.

36 Maciej Kurant, Athina Markopoulou, and Patrick Thiran. Towards unbiased BFS sampling. *IEEE Journal on Selected Areas in Communications*, 29(9):1799–1809, 2011. doi:10.1109/JSAC.2011.111005.

37 A. Lakhina, J.W. Byers, M. Crovella, and P. Xie. Sampling biases in IP topology measurements. In *IEEE INFOCOM*, volume 1, pages 332–341, 2003. doi:10.1109/INFCOM.2003.1208685.

38 Claire Mathieu and Hang Zhou. A simple algorithm for graph reconstruction. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 68:1–68:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.68.

39 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2/e edition, 2017.

40 Lev Reyzin and Nikhil Srivastava. On the longest path algorithm for reconstructing trees from distance matrices. *Inf. Process. Lett.*, 101(3):98–100, 2007. doi:10.1016/j.ipl.2006.08.013.

41 Guozhen Rong, Wenjun Li, Yongjie Yang, and Jianxin Wang. Reconstruction and verification of chordal graphs with a distance oracle. *Theoretical Computer Science*, 859:48–56, 2021. doi:10.1016/j.tcs.2021.01.006.

42 Sandeep Sen and V. N. Muralidhara. The covert set-cover problem with application to network discovery. In Md. Saidur Rahman and Satoshi Fujita, editors, *WALCOM: Algorithms and Computation, 4th International Workshop, WALCOM 2010, Dhaka, Bangladesh, February 10-12, 2010. Proceedings*, volume 5942 of *Lecture Notes in Computer Science*, pages 228–239. Springer, 2010. doi:10.1007/978-3-642-11440-3_21.

43 G. Tardos. Query complexity, or why is it difficult to separate $NP^A \cap coNP^A$ from P^A by random oracles A? *Combinatorica*, 9(4):385–392, December 1989. doi:10.1007/BF02125350.

44 Fabien Tarissan, Matthieu Latapy, and Christophe Prieur. Efficient measurement of complex networks using link queries. *CoRR*, abs/0904.3222, 2009. arXiv:0904.3222.

45 Mikkel Thorup and Uri Zwick. Compact routing schemes. In Arnold L. Rosenberg, editor, *13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001. doi:10.1145/378580.378581.

46 Zhaosen Wang and Jean Honorio. Reconstructing a bounded-degree directed tree using path queries. In *57th IEEE Allerton Conference on Communication, Control, and Computing*, 2019. See also arXiv:1606.05183.

47 Michael S Waterman, Temple F Smith, Mona Singh, and William A Beyer. Additive evolutionary trees. *Journal of theoretical Biology*, 64(2):199–213, 1977.

48 Andrew Chi-Chih Yao. Decision tree complexity and Betti numbers. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 615–624, New York, NY, USA, 1994. ACM. doi:10.1145/195058.195414.

49 X. Zhang and C. Phillips. A survey on selective routing topology inference through active probing. *IEEE Communications Surveys Tutorials*, 14(4):1129–1141, 2012.

50 Yaonan Zhang, Eric D. Kolaczyk, and Bruce D. Spencer. Estimating network degree distributions under sampling: An inverse problem, with applications to monitoring social media networks. *The Annals of Applied Statistics*, 9(1):166–199, 2015. doi:10.1214/14-AOAS800.

A Omitted Proofs

Here, we provide proofs that were omitted in the body of this paper.

A.1 Bounding the number of Bad Iterations for Parallel Graph Clustering

Recall that we chose α in our parallel graph clustering algorithm according to the formula

$$\alpha = \begin{cases} 2 & \text{if } \beta \le ((4/3)e)^4 \\ (4/3)e\beta^{1/2} & \text{else,} \end{cases}$$

and we said an iteration is "bad" if $\sum_{w\in W_i} |C_{A'_i}(w)| > \frac{\alpha n|W_i|}{s}$ and otherwise it is "good." We also noted, by Markov's inequality, that an iteration is bad with probability at most $1/\alpha$. Further, note that $L = \log_{(\beta/\alpha)} n = O(\log_\beta n)$ is the maximum number of good iterations, for either choice for α. We wish to show that the number of bad iterations is $O(L)$ w.h.p.

Since W_i is a given for iteration i, whether iteration i is good or bad depends only on A'_i; therefore, an iteration is good independent of whether any other iteration is good or bad, so, for the sake of analysis, consider a set of c_0L iterations (i.e., padding out with "dummy" iterations if necessary) where $c_0 \geq 4$ is a constant chosen below and each iteration is bad independently with probability $1/\alpha$. Let X denote the number of bad iterations in this set. So $E[X] = c_0L/\alpha$; hence, the probability that over $3/4$ of our iterations are bad can be bounded as $p = \Pr(X > (3/4)c_0L) = \Pr(X > (3/4)\alpha \cdot E[X])$. Thus, at least L of our iterations are good with probability at least $1 - p$.

Case 1: $\alpha = 2$. In this case, β is $O(1)$; hence, L is $\Theta(\log_\beta n) = \Theta(\log n)$, since $\beta \geq 4$. Futher, $\Pr(X > (3/4)\alpha \cdot E[X]) = \Pr(X > (3/2) \cdot E[X])$, and, by a standard Chernoff bound,[5] e.g., see [39, p. 69],

$$\Pr(X > (3/2) \cdot E[X]) \leq e^{-E[X]/12} = e^{-c_0L/24}.$$

Thus, choosing c_0 so that $c_0L/24 \geq 2\ln n$, we will have more than $(3/4)c_0L$ bad iterations with probability at most $1/n^2$.

Case 2: $\alpha = (4/3)e\beta^{1/2}$. In this case, $(\alpha/\beta) \leq \beta^{-1/4}$; hence, L is $O(\log_\beta n)$. Further, we have that $\Pr(X > (3/4)c_0L) = \Pr(X > (3/4)\alpha \cdot E[X]) = \Pr(X > e\beta^{1/2} \cdot E[X])$, and, by a non-standard Chernoff bound,[6] e.g., see [39, p. 70],

$$\Pr(X > e\beta^{1/2} \cdot E[X]) \leq \left(\frac{e}{e\beta^{1/2}}\right)^{(3/4)c_0L} = \beta^{-(3/8)c_0L}.$$

Thus, by choosing c_0 so that $(3/8)c_0L \geq 2\log_\beta n$, we will have more than $(3/4)c_0L$ bad iterations with probability at most $1/n^2$.

Therefore, we have the following.

▶ **Lemma 14.** *The number of good and bad iterations in Algorithm 1 is $O(\log_\beta n)$ w.h.p.*

A.2 The Complexity of the estimated-parallel-centers Algorithm

Recall that the estimated-parallel-centers algorithm uses a global random sample set, R, for estimating cluster sizes, where R is a random subset of U of size $T = c_1(s/\beta)\log n$. Recall that, for each vertex $w \in W$, we defined $S(w)$ such that $S(w) = R \cap C_A(w)$. Thus, $E[|S(w)|] = |C_A(w)|(T/n)$. We are interested in showing that w.h.p. this sample of $C_A(w)$ is giving neither an over-estimate nor an under-estimate for the size of $C_A(w)$, which we define respectively as follows:

- *Over-estimate event*: $|C_A(w)| \leq \beta n/s$, but $|S(w)| > 2\beta T/s$. In this case, we would be including w in W even though its cluster size is sufficiently small.
- *Under-estimate event*: $|C_A(w)| > 3\beta n/s$, but $|S(w)| \leq 2\beta T/s$. In this case, we would be excluding w from W even though its cluster size is big.

Let us consider each of these types of events in turn.

[5] $\Pr(X \geq (1+\delta) \cdot E[X]) \leq e^{-E[X]\cdot\delta^2/3}$, for $0 < \delta \leq 1$.
[6] $\Pr(X \geq (1+\delta) \cdot E[X]) \leq (e/(1+\delta))^{(1+\delta)\cdot E[X]}$.

Over-estimate event. We wish to bound the probability that $|C_A(w)| \le \beta n/s$ but $|S(w)| > 2\beta T/s$, where $T = c_1(s/\beta)\log n$. Let X denote the sum of $|C_A(w)|$ indicator random variables for counting the members of $C_A(w) \cap R$, i.e., where each variable is 1 independently with probability T/n. Thus, $E[X] = E[|S(w)|] = |C_A(w)|(T/n)$. So

$$\Pr(|S(w)| > 2\beta T/s) = \Pr(X > 2\beta T/s) = \Pr\left(X > \frac{2\beta n}{s|C_A(w)|} \cdot E[X]\right)$$
$$= \Pr(X > (1+\delta)\cdot E[X]),$$

where

$$\delta = \left(\frac{2\beta n}{s|C_A(w)|} - 1\right) \;>\; 1.$$

In addition,

$$\delta \cdot E[X] = \left(\frac{2\beta n}{s|C_A(w)|} - 1\right) \cdot \frac{|C_A(w)|T}{n} = \frac{2\beta T}{s} - \frac{|C_A(w)|T}{n}$$
$$\ge \frac{2\beta T}{s} - \frac{\beta T}{s} = \frac{\beta T}{s} = c_1 \log n.$$

Thus, by a standard Chernoff bound,[7] and the fact that $\delta > 1$,

$$\Pr(X \ge (1+\delta)\cdot E[X]) \le e^{-\delta^2 \cdot E[X]/(2+\delta)} \le e^{-\delta\cdot E[X]/3} \le e^{-(c_1 \log n)/3} \le \frac{1}{n^3},$$

for $c_1 \ge 9\ln 2 \approx 6.24$.

Under-estimate event. We wish to bound the probability that $|C_A(w)| > 3\beta n/s$ but $|S(w)| \le 2\beta T/s$, where $T = c_1(s/\beta)\log n$. Let X denote the sum of $|C_A(w)|$ indicator random variables for counting the members of $C_A(w) \cap R$, i.e., where each variable is 1 independently with probability T/n. Thus, $E[X] = E[|S(w)|] = |C_A(w)|(T/n) > 3c_1 \log n$. So

$$\Pr(|S(w)| \le 2\beta T/s) = \Pr(X \le 2\beta T/s)$$
$$= \Pr\left(X \le \frac{2\beta n}{s|C_A(w)|} \cdot E[X]\right) \le \Pr(X \le (2/3)\cdot E[X]).$$

Thus, by a standard Chernoff bound,[8] e.g., see [39, p. 71],

$$\Pr(|S(w)| \le 2\beta T/s) \le e^{-(3c_1 \log n)/18} \le \frac{1}{n^3},$$

when $c_1 \ge 18\ln 2 \approx 12.48$.

Of course, R is the same random sampling set for all our samples, $S(w)$, for $w \in W$. Nevertheless, by a union bound, the above analysis shows that R causes an over-estimate event or an under-estimate event, for some $S(w)$, with probability at most $1/n^2$.

By the bound on over-estimate events, we have shown that w.h.p. every cluster with size over $\beta n/s$ is included in W in any given iteration of our estimated-parallel-centers algorithm. In addition, by the bound on under-estimate events, we have shown that w.h.p. every vertex,

[7] $\Pr(X \ge (1+\delta)\cdot E[X]) \le e^{-\delta^2\cdot E[X]/(2+\delta)}$, for $\delta > 0$, e.g., see `https://en.wikipedia.org/wiki/Chernoff_bound`.

[8] $\Pr(X \le (1-\delta)\cdot E[X]) \le e^{-\delta^2\cdot E[X]/2}$, for $0 < \delta < 1$.

w, that we exclude from W has a cluster size of at most $3\beta n/s$. Thus, using essentially the same analysis as we gave for the proofs of Theorem 2 and Lemma 14, and noting that each iteration of our estimated-parallel-centers algorithm has round complexity $O(R(n))$ and query complexity $O(Q(n)(s/\beta)\log n)$, where $R(n)$ and $Q(n)$ are the respective round and query complexities for the Distances algorithm, we have the following.

▶ **Lemma 15** (Lemma 4). *Our estimated-parallel-centers algorithm constructs a set, A, of $(3\beta, s)$-balanced centers of size $O(s \log_\beta n)$. Suppose Distances(r, U) executes in $R(n)$ rounds and $Q(n)$ kth-hop queries. Then estimated-parallel-centers algorithm executes in $O(R(n)\log_\beta n)$ rounds and $O(Q(n)(s/\beta)\log n \, \log_\beta n)$ kth-hop queries, w.h.p.*

A.3 Greedy Algorithm Proofs

Our description of our greedy algorithm given above in the body of our paper was for the case when $U \subset V$. For the case when $U = V$, we modify our algorithm to be the following (note that in this case, hop-count distance and graph distance are the same):

1. We initialize a set of tentative edges, $\hat{E}$, to a spanning tree of H by calling kth-hop$(1, v, u)$ from every vertex, $v \in U$, to an arbitrarily chosen vertex, $u \in U$. We initialize a set of confirmed non-edges, $F \leftarrow \emptyset$. This requires 1 round of $O(n)$ kth-hop queries.
2. We compute all the $S_{u,v}(\hat{E})$ sets, for pairs $(u, v) \in \hat{E}^c$, which requires no queries.
3. We perform p steps of the greedy set-cover algorithm applied to the sets, $S_{u,v}(\hat{E})\backslash F$, with the goal of covering the remaining pairs, in $\hat{E}^c\backslash F$, in a greedy fashion, which also requires no queries. Let $\{(u_1, v_1), (u_2, v_2), \ldots, (u_p, v_p)\}$ denote the vertex pairs for the $S_{u,v}(\hat{E})$ sets chosen by these greedy steps.
4. We perform a binary search using kth-hop queries to determine the actual distance, $\delta(u_i, v_i)$, between u_i and v_i in H, for each $i = 1, 2, \ldots, p$ in parallel. This step requires $O(\log n)$ rounds of $O(p \log n)$ kth-hop queries in total.
5. For each i such that $\delta(u_i, v_i) = \hat{\delta}(u_i, v_i)$, we add all the pairs in $S_{u_i,v_i}(\hat{E})$ to F. If $F = \hat{E}^c$, then we are done, by Lemma 10.
6. Otherwise, if $F \neq \hat{E}^c$ and $\delta(u_i, v_i) = \hat{\delta}(u_i, v_i)$, for all $i = 1, 2, \ldots, p$, then we repeat the above process, performing another p steps of greedy set cover, repeating a loop returning to Step 3.
7. If, on the other hand, $F \neq \hat{E}^c$ and $\delta(u_i, v_i) < \hat{\delta}(u_i, v_i)$, for some i, then there must be at least one edge on a shortest path from u_i to v_i that is in E and not yet in $\hat{E}$. In this case, we perform a binary search, described below, to find at least one such an edge, add all such edges to $\hat{E}$, and repeat the above greedy searching for this updated set, $\hat{E}$, of candidate edges, returning to Step 2. This step requires $O(\log n)$ rounds of at most $O(p \log n)$ kth-hop queries in total.

Before we give our analysis, let us describe the details for the binary search to find an undiscovered edge when $\delta(u_i, v_i) < \hat{\delta}(u_i, v_i)$, for some i. We begin with a query, kth-hop(k, u_i, v_i), where $k = \lfloor \delta(u_i, v_i)/2 \rfloor$, and let w denote the returned vertex. So, $\delta(u_i, w) = k$ and $\delta(w, v_i) = \delta(u_i, v_i) - k$. Since $\delta(u_i, v_i) < \hat{\delta}(u_i, v_i)$, we know that $\delta(u_i, w) < \hat{\delta}(u_i, w)$ or $\delta(w, v_i) < \hat{\delta}(w, v_i)$. Thus, we recursively search for one of these until we discover a new edge not in $\hat{E}$, which must exist, since $\delta(u_i, v_i) < \hat{\delta}(u_i, v_i)$.

▶ **Theorem 16** (Theorem 11). *Let $f(n, \Delta)$ be the query complexity of an optimal sequential algorithm for graph verification for any unweighted connected graph with n vertices and maximum degree Δ using distance queries. Then, for $1 \le p < n$, our parallel network mapping algorithm has kth-hop query complexity, $Q(n) \in O((\Delta np + f(n, \Delta)\log n)\text{diam}(G))$*

and round complexity, $R(n) \in O((\Delta n + (f(n,\Delta)/p)\log n))$*, if* $U \subset V$*, or* $Q(n) \in O((\Delta np + f(n,\Delta)\log n)\log n)$ *and round complexity,* $R(n) \in O((\Delta n + (f(n,\Delta)/p)\log n)\log n)$*, if* $U = V$*.*

Proof. Building a spanning tree of H is a one-time expense taking $O(n \cdot \mathrm{diam}(G))$ kth-hop queries and a round complexity of $O(1)$ (step 1), for the case when $U \subset V$, or $O(n)$ queries with a round complexity of $O(1)$ (step 1), for the case when $U = V$. Each iteration of our greedy algorithm takes $O(p \cdot \mathrm{diam}(G))$ kth-hop queries with a round complexity of $O(1)$ (step 4), for the case when $U \subset V$, or $O(p \log n)$ kth-hop queries with a round complexity of $O(\log n)$ (step 4 and step 7), for the case when $U = V$.

In the case when $\delta_h(u_i, v_i) < \hat{\delta}_h(u_i, v_i)$, for some $i \in [1, p]$, we discover at least one new edge – let us charge the queries for this iteration to this edge. Thus, the total number of kth-hop queries due to this case is $O(\Delta np \cdot \mathrm{diam}(G))$, with $O(\Delta n)$ rounds, for the $U \subset V$ case, or $O(\Delta np \log n)$, with $O(\Delta n \log n)$ rounds, for the $U = V$ case. So, let us consider the case when $\delta_h(u_i, v_i) = \hat{\delta}_h(u_i, v_i)$, for all $i \in [1, p]$, which we call a "completely-greedy" iteration. We will provide an upper bound for the number of such iterations. Recall that in step 3, for the case when $U \subset V$ (similarly in step 3, for the case when $U = V$) we performed p steps of greedy set-cover algorithm applied to the sets, $S_{u,v}(\hat{E})\backslash F$, with the goal of covering the remaining pairs, in $\hat{E}^c\backslash F$ without additional queries. Let F_i denote the set of (x, y) pairs covered by the i-th step of this greedy set-cover algorithm, for $i = 1, 2, \ldots, p$. Thus,

$$|F_1| \geq |F_2| \geq \cdots \geq |F_p|,$$

and at the moment we chose the subset F_i it was the largest subset covering the uncovered pairs in $\mathcal{U}_i = \hat{E}^c\backslash(\bigcup_{j=1}^{i-1} F_j \cup F)$. The optimal sequential graph verification algorithm performs $f(n,\Delta)$ distance queries and confirms all the pairs in $\hat{E}^c$. Thus, in particular, this optimal algorithm must perform queries that cover $\mathcal{U}_i$ as a part of its $f(n,\Delta)$ queries; hence, because F_i is the subset for a distance query that covers the largest number of pairs in $\mathcal{U}_i$, and the average number of pairs in $\mathcal{U}_i$ covered by any distance query of the optimal algorithm is at least $|\mathcal{U}_i|/f(n,\Delta)$, we have that

$$|F_i| \geq \frac{|\mathcal{U}_i|}{f(n,\Delta)}.$$

Thus, in any iteration of our algorithm, since we perform p greedy steps, the size of the remaining pairs in $\hat{E}^c\backslash F$ is reduced by a multiplicative factor of

$$\left(1 - \frac{1}{f(n,\Delta)}\right)^p \leq e^{-p/f(n,\Delta)}.$$

Therefore, since $\hat{E}^c \leq n(n-1)$ and by the end of our algorithm we cover every pair in $\hat{E}^c$, the total number of completely-greedy iterations, g, can be bounded above by the smallest value of g such that

$$e^{-(p/f(n,\Delta))g} < n^{-2};$$

therefore, the total number of completely-greedy iterations, g, is at most $O((f(n,\Delta)/p)\log n)$. Note that the set $\hat{E}^c$ is potentially growing during our algorithm, with completely-greedy iterations possibly interspersed with iterations that discover new edges in $\tilde{E}$. Nevertheless, the above analysis still holds, because (1) the function, $f(n,\Delta)$ is a uniform bound for any connected graph with n nodes and maximum degree Δ, and (2) each time we (greedily) confirm that $\hat{\delta}(u,v) = \delta(u,v)$ for a set, $S_{u,v}(\hat{E})$, all the pairs in $S_{u,v}(\hat{E})$ are, in fact, non-edges in $\tilde{E}^c$. The claimed complexity bounds follow then, since each completely-greedy iteration requires $O(p \cdot \mathrm{diam}(G))$ kth-hop queries with round complexity $O(1)$, for the $U \subset V$ case, or $O(p \log n)$ kth-hop queries with round complexity $O(\log n)$, for the $U = V$ case. ◀

On Robustness for the Skolem and Positivity Problems

S. Akshay
Indian Institute of Technology Bombay, Mumbai, India

Hugo Bazille
Laboratoire de Recherche et Développement de l'Epita (LRDE), Rennes, France

Blaise Genest
Univ Rennes, CNRS, IRISA, France

Mihir Vahanwala
Indian Institute of Technology Bombay, Mumbai, India

Abstract

The Skolem problem is a long-standing open problem in linear dynamical systems: can a linear recurrence sequence (LRS) ever reach 0 from a given initial configuration? Similarly, the positivity problem asks whether the LRS stays positive from an initial configuration. Deciding Skolem (or positivity) has been open for half a century: The best known decidability results are for LRS with special properties (e.g., low order recurrences). On the other hand, these problems are much easier for "uninitialized" variants, where the initial configuration is not fixed but can vary arbitrarily: checking if there is an initial configuration from which the LRS stays positive can be decided by polynomial time algorithms (Tiwari in 2004, Braverman in 2006).

In this paper, we consider problems that lie between the initialized and uninitialized variant. More precisely, we ask if 0 (resp. negative numbers) can be avoided from every initial configuration in a neighborhood of a given initial configuration. This can be considered as a robust variant of the Skolem (resp. positivity) problem. We show that these problems lie at the frontier of decidability: if the neighborhood is given as part of the input, then robust Skolem and robust positivity are Diophantine-hard, i.e., solving either would entail major breakthrough in Diophantine approximations, as happens for (non-robust) positivity. Interestingly, this is the first Diophantine-hardness result on a variant of the Skolem problem, to the best of our knowledge. On the other hand, if one asks whether such a neighborhood exists, then the problems turn out to be decidable in their full generality, with PSPACE complexity. Our analysis is based on the set of initial configurations such that positivity holds, which leads to new insights into these difficult problems, and interesting geometrical interpretations.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Skolem problem, verification, dynamical systems, robustness

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.5

Funding This work was partly supported by the DST/CEFIPRA/INRIA associated team EQuaVE, DST/SERB Matrices grant MTR/2018/00074, and ANR-20-CE25-0012 MAVeriQ.

1 Introduction

A linear recurrence relation (LRR) is a relation $u_{n+\kappa} = \sum_{j=0}^{\kappa-1} a_j \cdot u_{n+j}$ for all $n, \kappa \in \mathbb{N}, \kappa \geq 1$, defined by a tuple of non-negative, rational coefficients $(a_0, \dots, a_{\kappa-1})$. Given the first κ entries of the recurrence $u_0, \dots u_{\kappa-1}$ (called the initial configuration), the LRR uniquely defines an infinite sequence $(u_n)_{n\in\mathbb{N}}$, called a Linear Recurrence Sequence (LRS). The Skolem problem asks, given an LRS, i.e., a recurrence relation and an initial configuration, whether the sequence ever hits 0, i.e. does there exist $n \in \mathbb{N}$ with $u_n = 0$. The positivity problem is a

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 5; pp. 5:1–5:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

variant where the question asked is whether for all $n \in \mathbb{N}$, $u_n \geq 0$. Both these problems have applications in software verification, probabilistic model checking, discrete dynamic systems, theoretical biology, economics.

While the statements seem simple, the decidability of these problems remains open since their introduction in the 1930's. Only partial decidability results are known, e.g., when the dimension is <5 [29]. For a subclass of LRS called simple, positivity is decidable for order <10 [23]. On top of the inability to provide an algorithm to decide Skolem or positivity in the general case, the authors of [24] prove an important hardness result: solving positivity would entail a major breakthroughs in Diophantine approximations. More precisely, one would be able to approximate the type of many transcendental numbers t, i.e., how close one can approximate t with rational numbers with small denominators.

This hardness result contrasts with positive results obtained for relaxations of the problems. First, the continuous relaxation, where instead of considering discrete steps for the recurrence, Chonev et al [13] considers a continuous process, and some corresponding questions turn out to be decidable subject to Schanuel's Conjecture. Second, instead of considering a fixed initial configuration, [28, 12] consider every possible configuration as initial, i.e., they ask if there exists an initial configuration starting from which ensures that all entries of the sequence remain positive (this is sometimes called the uninitialized positivity problem). Surprisingly they show that this problem can be decided in PTIME. More recently, this result has been extended to processes with choices [5].

In this paper, we consider a natural variant that lies between the hard question of fixed initial configuration [24], and the easy question when the initial configuration is totally unconstrained [28, 12]. More precisely, we ask whether starting from an initial configuration in a neighborhood, all entries of the recurrence sequence remain positive (we call this the *robust positivity problem*) or away from zero (we call this the *robust Skolem problem*). An immediate question that arises is whether the neighborhood is part of the input or not and it turns out that this has a significant impact on decidability, as we discuss next. Our motivation to look at these problems comes from their role in capturing a powerful and natural notion of robustness, where the exact initial configuration cannot be fixed with arbitrarily high precision (which is often the case with real systems).

Since we need to tackle multiple initial configurations, we reason about the set of initial configurations from which positivity holds, which is sufficient to answer robustness questions. For that, we revisit the usual algebraic equations in a more graphical manner, which forms the crux of our approach. This allows us to reinterpret and generalize the hardness result of [23], giving our first main contribution: if the neighborhood is given as a fixed ball, then the problems remain hard: both robust Skolem and robust positivity are Diophantine-hard. Interestingly, this holds regardless of whether the ball is open or closed.

We then turn to the problems where the ball is not fixed, and ask if there exists a radius $\psi > 0$ such that 0 or negative numbers can be avoided from every initial configuration in the ψ ball around a given initial configuration. Our second main contribution is to show that this robust version of the Skolem and positivity problems are both decidable in full generality, with PSPACE complexities.

Related work. As mentioned earlier, the Skolem problem and its variants have received a lot of attention. Given the hardness of these problems, ε-approximate solutions have been considered, e.g., in [9, 1] with different definitions of approximations. In comparison with our work, these are designed towards allowing approximate model checking. More recently the notion of imprecision in Skolem and related problems was considered in [6, 15]. In [6], the authors consider rounding functions at every step of the trajectory. In [15], the so called Pseudo-Skolem problem is defined, where imprecisions up to ε are allowed at every step of the trajectory, which is shown to be decidable in PTIME. These are quite different from our notion of robustness, which faithfully considers the trajectories generated from a ball representing ε-perturbations around the initial configuration. Lastly, [22] considers real numbers as input (instead of rational numbers). This allows one to consider the set of initial configurations for which decidability of Skolem is not known, and show that this set has Lebesgue measure 0.

2 Preliminaries

Let κ be any non-negative integer (which will be used to denote the order of the LRS). Let $\mathbf{c}, \mathbf{d}$ be two vectors of $\mathbb{R}^\kappa$ that can be seen as one dimensional matrices of $\mathbb{R}^{\kappa\times 1}$. The distance between $\mathbf{c}, \mathbf{d}$ is defined as $||\mathbf{c}-\mathbf{d}|| = \sqrt{(\mathbf{c}-\mathbf{d})^T(\mathbf{c}-\mathbf{d})}$, the standard L_2 distance. In this paper, we will consider two norms on vectors: the first is the standard L_2 norm $||\mathbf{c}||$. The second is size($\mathbf{c}$), denoting the size of its bit representation i.e., number of bits needed to write down $\mathbf{c}$ (for complexity). We use the same notation for scalar constants with size(a) denoting the number of bits to represent a real/rational constant a. An algebraic number α is a root of a polynomial p with integer coefficients. It can be represented uniquely [20] by a 4-tuple (p, a, b, r) as the only root of p at distance $< r$ of $a + ib$, with $a, b, r \in \mathbb{Q}$ (also see Appendix A.1). We define $size(\alpha)$ as the size of the bit representation of (p, a, b, r).

2.1 Linear Recurrence Sequences

We start by defining linear recurrence relations and sequences over rationals.

▶ **Definition 1.** *A linear recurrence relation $(u_n)_{n\in\mathbb{N}}$ of order κ is specified by a tuple of coefficients $\mathbf{a} = (a_0, \dots, a_{\kappa-1}) \in \mathbb{Q}^\kappa$. Given an initial configuration $\mathbf{c} = (c_0, \dots, c_{\kappa-1}) \in \mathbb{Q}^\kappa$, the LRR uniquely defines a linear recurrence sequence (LRS henceforth), which is the sequence $(u_n(\mathbf{c}))_{n\in\mathbb{N}}$, inductively defined as $u_j(\mathbf{c}) = c_j$ for $j \le \kappa - 1$, and*

$$u_{n+\kappa}(\mathbf{c}) = \sum_{j=0}^{\kappa-1} a_j u_{n+j}(\mathbf{c}) \text{ for all } n \in \mathbb{N}.$$

The companion matrix associated with the LRR/LRS (it does not depend upon the initial configuration $\mathbf{c}$) is:

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ a_0 & a_1 & a_2 & \dots & a_{\kappa-1} \end{bmatrix}.$$

The characteristic polynomial of the LRR/LRS is $X^\kappa - \sum_{j=0}^{\kappa-1} a_j X^j$. The LRS is said to be simple if every root of the characteristic polynomial has multiplicity one. The size s of the LRS is the size of its bit representation and is given by $s = \sum_{j=0}^{\kappa-1}(size(a_j) + size(c_j))$.

Notice that given an initial configuration $\mathbf{c} \in \mathbb{Q}^\kappa$, we have that $\mathbf{M}^n\mathbf{c} = (u_n(\mathbf{c}), \dots, u_{n+\kappa-1}(\mathbf{c}))$. Reasoning in the κ dimensions $(u_n, \dots, u_{n+\kappa-1})$ is a very useful technique that we will use throughout the paper as it displays the LRR as a linear transformation $\mathbf{M}$.

The characteristic roots of an LRR/LRS are the roots of its characteristic polynomial, and also the eigenvalues of the companion matrix. Let $\gamma_1, \dots, \gamma_r \in \mathbb{C}$ be the characteristic roots of the LRR/LRS. An eigenvalue γ_i is called *dominant* if it has maximal modulus $|\gamma_i| = \max_{j \leq r} |\gamma_j|$, and *residual* otherwise. For all $j \leq r$, γ_j is algebraic and $\text{size}(\gamma_j) = s^{\mathcal{O}(1)}$. We denote by m_j the multiplicity of γ_j. We have $\sum_{j=1}^{r} m_j = \kappa$.

▶ **Proposition 2** (Exponential polynomial solution [16]). *Given an initial configuration* $\mathbf{c}$*, there exists a unique tuple of coefficients* $(\alpha_{ij}(\mathbf{c}))_{i \leq r, j < m_r}$ *such that for all* n*,*

$$u_n(\mathbf{c}) = \sum_{i=1}^{r} \left(\sum_{j=0}^{m_r - 1} \alpha_{ij}(\mathbf{c}) n^j \right) \gamma_i^n.$$

The coefficients $\alpha_{ij}(\mathbf{c})$ can be solved for from the initial state $\mathbf{c}$ [17]. It is implicit in the solution that for all i, j, both α_{ij} and $\frac{1}{\alpha_{ij}}$ are algebraic with values and norms upper bounded by $2^{s^{\mathcal{O}(1)}}$. A formal proof of this claim can be found in [2, Lemmas 4, 5, 6].

If the LRS is simple, then by definition $m_i = 1$ for all i, and $u_n = \sum_{i=1}^{r} \alpha_i(\mathbf{c}) \gamma_i^n$, with $\alpha_i(\mathbf{c})$ linear in $\mathbf{c}$, ie $\alpha_i(\lambda \mathbf{c} + \lambda' \mathbf{c}') = \lambda \alpha_i(\mathbf{c}) + \lambda' \alpha_i(\mathbf{c}')$.

▶ **Example 3.** Consider the Linear Recurrence Relation of order 6 with $\mathbf{a} = (-1, 4, -8, 10, -8, 4)$, i.e. $u_{n+6} = 4u_{n+5} - 8u_{n+4} + 10u_{n+3} - 8u_{n+2} + 4u_{n+1} - u_n$. The roots of the characteristic polynomial are $1, e^{i2\pi\theta}, e^{-i2\pi\theta}$, with $\theta = \frac{1}{3}$, each with multiplicity 2, and all dominant (they have the same modulus 1). The exponential polynomial solution is of the form $u_n(\mathbf{c}) = z(\mathbf{c})n + z'(\mathbf{c}) + (x(\mathbf{c})n + x'(\mathbf{c}))e^{i2\pi n\theta} + (y(\mathbf{c})n + y'(\mathbf{c}))e^{-i2\pi n\theta}$. As $u_n(\mathbf{c})$ is real, we must have that $x(\mathbf{c}), y(\mathbf{c})$ are conjugates, as well as $x'(\mathbf{c}), y'(\mathbf{c})$, and thus:

$$u_n(\mathbf{c}) = z(\mathbf{c})n + z'(\mathbf{c}) + 2(Re(x(\mathbf{c}))n + Re(x'(\mathbf{c}))) \cos(2\pi n\theta) + 2(Im(x(\mathbf{c}))n + Im(x'(\mathbf{c}))) \sin(2\pi n\theta).$$

2.2 Skolem and positivity problems

▶ **Definition 4** (Skolem problem). *Let* $(u_n)_{n \in \mathbb{N}}$ *an LRR and* $\mathbf{c} \in \mathbb{Q}^{\kappa}$*. The Skolem problem is to determine if there exists* $n \in \mathbb{N}$ *such that* $u_n(\mathbf{c}) = 0$*. The positivity (resp. strict positivity) problem is to determine if for all* $n \in \mathbb{N}$*,* $u_n(\mathbf{c}) \geq 0$ *(resp.* $u_n(\mathbf{c}) > 0$*).*

In this work, we will be more interested in the complement problem of Skolem: namely, whether $u_n(\mathbf{c}) \neq 0$ for all n. This is of course equivalent in terms of decidability, but this formulation is more meaningful in terms of robustness, where we want to robustly avoid 0.

The famous Skolem-Mahler-Lech theorem states that the set $\{i \mid u_i(\mathbf{c}) = 0\}$ is the union of a finite set F and finitely many arithmetic progressions [27, 18, 8]. These arithmetic progressions can be computed but the hard part lies in deciding if the set F is empty: although we know that there is N such that for all $n > N$, $n \notin F$, we do not have an effective bound on this N in general. The Skolem problem has been shown to be decidable for LRS of order up to 4 [21, 29] and is still open for LRS of higher order. Also, only an NP-hardness bound is known if the order is unrestricted [10, 3].

For simple LRS, positivity has been shown to be decidable up to order 9 [23]. In [25], it is proved that positivity for simple LRS is hard for co$\exists\mathbb{R}$, the class of problems whose complements are solvable in the existential theory of the reals. A last result, from [24], shows the difficulty of positivity, linking it to Diophantine approximation: how close one can approximate a transcendental number with a rational number with small denominator. We will follow the reasoning from [24]. The *Diophantine approximation type* of a real number x is defined as:

$$L(x) = \inf \left\{ c \in \mathbb{R} \mid \left| x - \frac{n}{m} \right| < \frac{c}{m^2},\ n, m \in \mathbb{Z} \right\}.$$

As mentioned in [24], the Diophantine approximation type of most transcendental numbers is unknown. Let $\mathcal{A} = \{p + qi \in \mathbb{C} \mid p, q \in \mathbb{Q} \setminus \{0\}, p^2 + q^2 = 1\}$, i.e., the set of points on the unit circle of $\mathbb{C}$ with rational real and imaginary parts, excluding $1, -1, i$ and $-i$. The set $\mathcal{A}$ consists of algebraic numbers of degree 2, none of which are roots of unity [24]. In particular, writing $p + qi = 2^{i2\pi\theta} = (-1)^{2\theta}$, we have that $\theta \notin \mathbb{Q}$ [24]. We denote:

$$\mathcal{T} = \left\{\theta \in (-1/2, 1/2] \mid e^{2\pi i\theta} \in \mathcal{A}\right\}.$$

As argued in [24], the set $\mathcal{T}$ is dense in $(-\frac{1}{2}, \frac{1}{2}]$, and is made only of transcendental numbers. In general, we don't have a method to compute $L(\theta)$ for $\theta \in \mathcal{T}$ (or approximate it with arbitrary precision):

▶ **Definition 5.** *We say that a problem is $\mathcal{T}$-Diophantine hard if its decidability entails that for all $\theta \in \mathcal{T}$ and $\varepsilon > 0$, one can compute a number ℓ such that $|\ell - L(\theta)| \leq \varepsilon$.*

Remarkably, in [24], it is shown that if one can solve the positivity problem in general, then one can also approximate $L(\theta)$. That is,

▶ **Theorem 6** ([24])**.** *Positivity for LRS of order 6 is $\mathcal{T}$-Diophantine hard.*

3 Robust Skolem and Robust Positivity

Both Skolem and Positivity consider a single initial configuration $\mathbf{c}$. In this article, we investigate the notion of robustness, that is, whether the property is true in a neighborhood of $\mathbf{c}$, which is important for real systems, where setting $\mathbf{c}$ with an arbitrary precision is not possible. We will consider two variants. The first one fixes the neighborhood as a ball $\mathcal{B}_\psi$ of radius $\psi > 0$ around an initial configuration $\mathbf{c}$, while the second one asks for the existence of an $\psi > 0$ such that for every initial configuration in $\mathcal{B}_\psi$, the respective condition is satisfied.

▶ **Definition 7** (Robustness for Skolem and Positivity)**.** *Let $(u_n)_{n\in\mathbb{N}}$ be a linear recurrence relation (specified by the coefficient $\mathbf{a} \in \mathbb{Q}^\kappa$), and $\mathbf{c} \in \mathbb{Q}^\kappa$ an initial configuration.*

Given $\psi > 0$, the robust Skolem *(resp.* robust positivity*) problem is to determine if for all $\mathbf{c}'$ with $||\mathbf{c}' - \mathbf{c}|| < \psi$ (open balls), or $||\mathbf{c}' - \mathbf{c}|| \leq \psi$ (closed balls), we have $u_n(\mathbf{c}') \neq 0$ (resp. $u_n(\mathbf{c}') \geq 0$) for all $n \in \mathbb{N}$.*

The $\exists$-robust Skolem *(resp.* $\exists$-robust positivity*) problem is to determine if there exists $\psi > 0$ such that for all $||\mathbf{c}' - \mathbf{c}|| < \psi$ we have $u_n(\mathbf{c}') \neq 0$ (resp. $u_n(\mathbf{c}') \geq 0$) for all $n \in \mathbb{N}$.*

Notice that we do not consider explicitly the case of closed balls for $\exists$-robust Skolem (resp. positivity), because there exists an open ball of radius $\psi > 0$ for which robust Skolem (resp. positivity) holds iff there exists a closed ball of radius $\psi' > 0$ (e.g. $\psi' = \frac{\psi}{2}$) for which it holds.

Our main results investigate the decidability and complexity of these problems.

▶ **Theorem 8.** *Robust Skolem and robust positivity are $\mathcal{T}$-Diophantine hard, even restricted to recurrence relations of order 6 for open or closed balls of rational radius ψ.*

Our first result means that uninitialized positivity really needs the initial configuration to take a value possibly anywhere in the space rather than in a fixed neighborhood to obtain decidability via [28, 12]. We remark that Diophantine-hardness is known for the non-robust variant of positivity [24], but to the best of our knowledge, it was not known for any variant of the Skolem problem.

Surprisingly, by relaxing the neighborhood to be as small as desired, one obtains decidability in full generality, as stated by our second main result:

▶ **Theorem 9.** *$\exists$-robust Skolem and $\exists$-robust positivity are decidable in* PSPACE.

The main difference between our techniques and several past work (except [4] which is restricted to eigenvalues being roots of unity) is as follows: given an LRR $(u_n)_{n\in\mathbb{N}}$, our intuition and proofs hinge on representing the set P of initial configurations $\mathbf{d}$ from which positivity holds. Formally:

$$P = \{\mathbf{d} \in \mathbb{R}^k \mid u_n(\mathbf{d}) \geq 0 \text{ for all } n \in \mathbb{N}\}.$$

We may note that the set P is convex. To see this, observe that for $d, d' \in P$, for all $\alpha, \beta > 0$ with $\alpha + \beta = 1$, we have $\alpha\mathbf{d} + \beta\mathbf{d}' \in P$ as $u_n(\alpha\mathbf{d} + \beta\mathbf{d}') = \alpha u_n(\mathbf{d}) + \beta u_n(\mathbf{d}') \geq 0$ for all n. We also remark that a definition similar to P is possible for the set S of initial configurations from which 0 is avoided. But it turns out that that set is much harder to represent (e.g., it is not convex in general). Using P surprisingly suffices to deal with robust Skolem as well.

In Section 4, we provide the geometric intuitions behind our ideas as well as set up the notations for the proofs of the above theorems. We exploit the geometric intuitions from Section 4 in Section 5, to prove Theorem 8 and in Section 6, to prove Theorem 9.

4 Geometrical representation of an LRR for Diophantine-hardness

We will show that, as for the non-robust variant, hardness starts at order 6. Hence, in this section and the next, we will focus on a particular LRR of order $\kappa = 6$, sufficient for the proof of hardness, i.e. Theorem 8. In Section 6, we will generalize some of the constructions explored here to obtain decidability of $\exists$-robust Skolem.

Let $\theta \in \mathcal{T}$, i.e. $e^{i2\pi\theta} = p + qi \in \mathcal{A}$, with both p, q rational and $p^2 + q^2 = 1$. We want to approximate $L(\theta)$ (indeed this is the problem that is "Diophantine-hard"). Consider the Linear Recurrence Relation of order 6 defined by $\mathbf{a} = (-1, 4p+2, -(4p^2+8p+3), 8p^2+8p+4, -(4p^2+8p+2), 4p+2)$. The roots of the characteristic polynomial are $1, e^{i2\pi\theta}, e^{-i2\pi\theta}$, each with multiplicity 2, and all dominant (they have the same modulus 1). Example 3 is a particular case of this $\mathbf{a}$, with $p = \frac{1}{2} = \cos(\frac{\pi}{3})$. However, notice that $\theta = \frac{1}{3} \notin \mathcal{T}$ as it corresponds to $q = \sin(\frac{\pi}{3}) = \frac{\sqrt{3}}{2} \notin \mathbb{Q}$. Now, since $u_n(\mathbf{c})$ is a real number for any n and real initial configuration $\mathbf{c}$, we can write the exponential polynomial solution in the form:

$$\begin{aligned} u_n(\mathbf{c}) =& z_{dom}(\mathbf{c})n - x_{dom}(\mathbf{c})n\cos(2\pi n\theta) - y_{dom}(\mathbf{c})n\sin(2\pi n\theta) \\ &+ z_{res}(\mathbf{c}) - x_{res}(\mathbf{c})\cos(2\pi n\theta) - y_{res}(\mathbf{c})\sin(2\pi n\theta). \end{aligned}$$

The coefficients $z_{dom}(\mathbf{c}), x_{dom}(\mathbf{c}), y_{dom}(\mathbf{c})$ and $z_{res}(\mathbf{c}), x_{res}(\mathbf{c}), y_{res}(\mathbf{c})$ are associated with the initial configuration $\mathbf{c}$ of the LRS. In the following, we reason in the basis of vectors $\overrightarrow{z_{dom}}, \overrightarrow{x_{dom}}, \overrightarrow{y_{dom}}, \overrightarrow{z_{res}}, \overrightarrow{x_{res}}, \overrightarrow{y_{res}}$, as the geometrical interpretation is simpler in this basis. We will eventually get back to the original coordinate vector basis at the end of the process. From e.g., [17, Section 2], we know that we can transform from one basis to the other using an invertible Matrix C with $C \cdot \mathbf{c} = (z_{dom}(\mathbf{c}), x_{dom}(\mathbf{c}), y_{dom}(\mathbf{c}), z_{res}(\mathbf{c}), x_{res}(\mathbf{c}), y_{res}(\mathbf{c}))$.

We study the positivity of u_n by studying the positivity of $v_n = \frac{u_n}{n}$, for all $n \geq 1$. We denote $v_n^{dom}(z_{dom}, x_{dom}, y_{dom}) = z_{dom} - x_{dom}\cos(2\pi n\theta) - y_{dom}\sin(2\pi n\theta)$, which we call the dominant part of v_n, while we denote $v_n^{res}(z_{res}, x_{res}, y_{res}) = \frac{1}{n}(z_{res} - x_{res}\cos(2\pi n\theta) - y_{res}\sin(2\pi n\theta))$, which we call the residual part of v_n. The residual part tends towards 0 when n tends towards infinity because of the coefficient $\frac{1}{n}$.

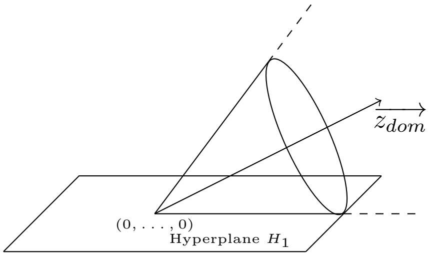


Figure 1 Visual representation of the cone $P_{(0,0,0)}$.

4.1 High-Level intuition and Geometrical Interpretation

We provide a geometrical interpretation of set P. We cannot characterize it exactly, even in this particular LRR of order $\kappa = 6$ (else we could decide positivity for this case which is known to be Diophantine hard). To describe P, we define its "section" over $(z_{dom}, x_{dom}, y_{dom})$ given $(z_{res}, x_{res}, y_{res})$:

$$P_{(z_{res},x_{res},y_{res})} = \{(z_{dom}, x_{dom}, y_{dom}) \mid v_n(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res}) \geq 0 \text{ for all } n\}.$$

It suffices to characterize $P_{(z_{res},x_{res},y_{res})}$ for all $(z_{res}, x_{res}, y_{res})$ in order to characterize P, as $P = \{(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res}) \mid (z_{dom}, x_{dom}, y_{dom}) \in P_{(z_{res},x_{res},y_{res})}\}$. Among these sets, one is particularly interesting: $P_{(0,0,0)}$, as it is the set of tuples $(z_{dom}, x_{dom}, y_{dom})$ such that $v_n^{dom}(z_{dom}, x_{dom}, y_{dom}) \geq 0$ for all $n \in \mathbb{N}$. Our reason for focussing on this representation of P is three-fold. First, unlike P, the set $P_{(0,0,0)}$ can be characterized exactly, as a cone depicted in Figure 1 (this will be formally shown in Lemma 10 below). Second, the set $P_{(z_{res},x_{res},y_{res})}$ is in 3 dimensions that we can represent more intuitively than a 6 dimensional set. Last but not least, we can show that $P_{(z_{res},x_{res},y_{res})} \subseteq P_{(0,0,0)}$ for all $(z_{res}, x_{res}, y_{res})$ (Lemma 12).

On the other hand, we also consider a related set in 6 dimensions:

$$P_{dom} = \{(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res}) \mid \forall n, v_n^{dom}(z_{dom}, x_{dom}, y_{dom})) \geq 0\}.$$

We note that $P_{(0,0,0)}$ is the projection of P_{dom} over the 3 dimensions $(z_{dom}, x_{dom}, y_{dom})$. Also, characterizing $P_{(0,0,0)}$ is sufficient to characterize P_{dom} as $(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res}) \in P_{dom}$ iff $(z_{dom}, x_{dom}, y_{dom}) \in P_{(0,0,0)}$. As $P_{(z_{res},x_{res},y_{res})} \subseteq P_{(0,0,0)}$ for all $(z_{res}, x_{res}, y_{res})$, we have $P \subseteq P_{dom}$.

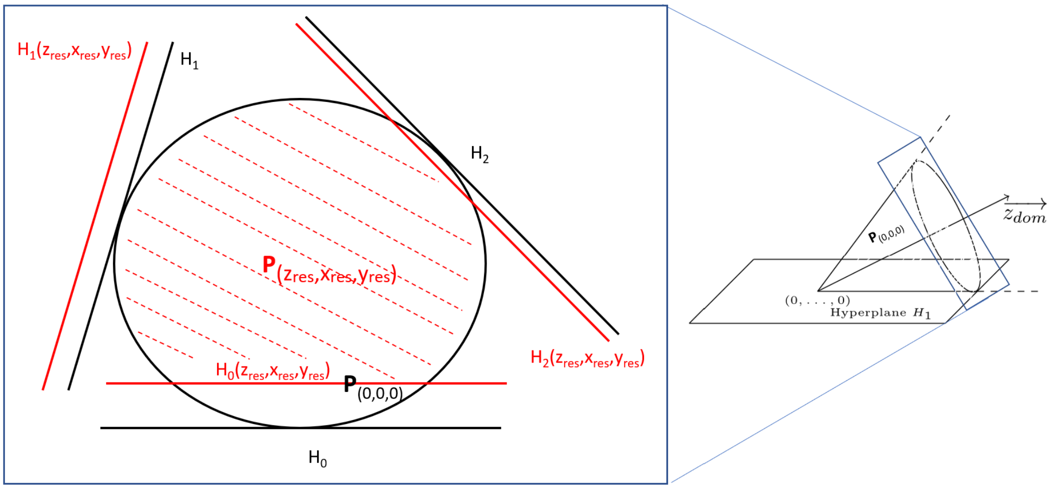
We are now ready to represent $P_{(z_{res},x_{res},y_{res})}$ given some value $(z_{res}, x_{res}, y_{res})$. We can interpret $P_{(z_{res},x_{res},y_{res})}$ in terms of half spaces: $P_{(z_{res},x_{res},y_{res})} = \bigcap_{m=1}^{\infty} H_m^+(z_{res}, x_{res}, y_{res})$, with $H_m^+(z_{res}, x_{res}, y_{res}) = \{(z_{dom}, x_{dom}, y_{dom}) \mid v_m(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res})) \geq 0\}$. The half space $H_m^+(z_{res}, x_{res}, y_{res})$ is delimited by hyperplane

$$H_m(z_{res}, x_{res}, y_{res}) = \{(z_{dom}, x_{dom}, y_{dom}) \mid v_m(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res})) = 0\}$$

which is a vector space ($\cos(2\pi m\theta)$ and $\sin(2\pi m\theta)$ are constant when m is fixed).

Consider the case of $(z_{res}, x_{res}, y_{res}) = (0, 0, 0)$. We denote $H_m^+ = H_m^+(0, 0, 0)$ and $H_m = H_m(0, 0, 0)$ for all m. For instance, $H_0 = \{(z_{dom}, x_{dom}, y_{dom}) \mid z_{dom} = x_{dom}\}$, as $v_0^{dom}(z_{dom}, x_{dom}, y_{dom}) = z_{dom} - x_{dom}$.

Let $\mathbf{M}_{dom}$ be the matrix associated with LRS $(v_n^{dom})_{n\in\mathbb{N}}$. We have $H_m = \mathbf{M}_{dom} H_{m-1} = \mathbf{M}_{dom}^m H_0$. We characterize $\mathbf{M}_{dom}$ in Lemma 11 as a rotation around $\overrightarrow{z_{dom}}$ of angle $-2\pi\theta$, which allows to characterize H_m as the hyperplane which is the rotation of H_0 of angle $2m\pi\theta$ around $\overrightarrow{z_{dom}}$. That is, the cone shape for $P_{(0,0,0)}$is obtained by cutting away chunk of the 3D space delimited by hyperplanes (H_m), the rotation $2n\pi\theta$ being dense in $[-\pi, \pi]$.


$H_1(z_{res}, x_{res}, y_{res})$
H_1
H_2
$P_{(z_{res}, x_{res}, y_{res})}$
$H_0(z_{res}, x_{res}, y_{res})$
$P_{(0,0,0)}$
$H_2(z_{res}, x_{res}, y_{res})$
H_0
$P_{(0,0,0)}$
z_{dom}
$(0, \ldots, 0)$
Hyperplane H_1

■ **Figure 2** Sections of $P_{(0,0,0)}$ (in black) and $P_{(z_{res},x_{res},y_{res})}$ (in dashed red), carved out by hyperplanes (H_i) (in black) and $(H_i(z_{res}, x_{res}, y_{res}))$ (in red) respectively.

Coming back to some value $(z_{res}, x_{res}, y_{res}) \neq (0, 0, 0)$, we have that the hyperplane $H_n(z_{res}, x_{res}, y_{res})$ is parallel to the hyperplane H_n (which is tangent to the cone $P_{(0,0,0)}$), because for H_n of the form $uz_{dom}+vx_{dom}+wy_{dom} = 0$, we have $H_n(z_{res}, x_{res}, y_{res})$ is defined by $\{(z_{dom}, x_{dom}, y_{dom}) \mid uz_{dom}+vx_{dom}+wy_{dom} = C\}$, for $C = \frac{z_{res}+x_{res}\cos(2\pi n\theta)+y_{res}\sin(2\pi n\theta)}{n}$ a constant as n is fixed.

Thus, with this idea in mind, we can visualize $P_{(z_{res},x_{res},y_{res})}$ as depicted in Figure 2, using $P_{(0,0,0)}$ and the hyperplanes $H_n(z_{res}, x_{res}, y_{res})$ parallel to H_n, with an explicit bound on the distance from $H_n(z_{res}, x_{res}, y_{res})$ to H_n, which further tends towards 0 as n tends towards infinity. Next, we formalize the above intuition/picture into lemmas.

4.2 Characterization of $P_{(0,0,0)}$ and representing $P_{(z_{res},x_{res},y_{res})}$

We now formalize some of the ideas in the above subsection. First, we start with Lemma 10 which shows that $P_{(0,0,0)}$ describes a cone, as displayed on Figure 1.

▶ **Lemma 10.** $P_{(0,0,0)} = \{(z_{dom}, x_{dom}, y_{dom}) \mid z_{dom} \geq \sqrt{x_{dom}^2 + y_{dom}^2}\}$.

Proof. We have $\cos(2\pi n\theta)^2 + \sin(2\pi n\theta)^2 = 1$ and $\cos(2\pi n\theta)$ is dense in $[-1, 1]$ as $\theta \notin \mathbb{Q}$. Denote $X = \cos(2\pi n\theta)$, and study the function $f(X) = x_{dom}X + y_{dom}\sqrt{1 - X^2}$. Its derivative is $f'(X) = x_{dom} - \frac{y_{dom}X}{\sqrt{1-X}\sqrt{1+X}}$. We have $f'(X) = 0$ iff $X = X_0 = \frac{x_{dom}}{\sqrt{x_{dom}^2+y_{dom}^2}}$. This gives a maximum for $f(X_0) = \frac{x_{dom}^2+y_{dom}^2}{\sqrt{x_{dom}^2+y_{dom}^2}} = \sqrt{x_{dom}^2 + y_{dom}^2}$. Thus, for all $(z_{dom}, x_{dom}, y_{dom})$ with $z_{dom} \geq \sqrt{x_{dom}^2 + y_{dom}^2}$, we have $z_{dom} \geq \max(f(X))$ and $v_n((z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res}) \geq z_{dom} - f(X) \geq 0$ for all n. On the other hand, if $z_{dom} < \sqrt{x_{dom}^2 + y_{dom}^2}$, then there exists n such that $f(\cos(2\pi n\theta))$ is arbitrarily close to $\max f(X) > z_{dom}$, and in particular $v_n = z_{dom} - f(\cos(2\pi n\theta)) < 0$. ◀

We show in Appendix A.2 the following lemma which states the linear function $\mathbf{M}_{dom}$ associated with the LRR $(v_n^{dom})_{n\in\mathbb{N}}$ is actually a rotation of angle $-2\pi\theta$.

▶ **Lemma 11.** $\mathbf{M}_{dom}(z_{dom}, x_{dom}, y_{dom}) = (z_{dom}, x_{dom}\cos(2\pi\theta) + y_{dom}\sin(2\pi\theta), y_{dom}\cos(2\pi\theta) - x_{dom}\sin(2\pi\theta))$, *that is* $\mathbf{M}_{dom}$ *is a rotation around axis* $\overrightarrow{z}$ *of angle* $-2\pi\theta$.

Finally, the following lemma implies that $P \subseteq P_{dom}$.

▶ **Lemma 12.** *For all $z_{res}, x_{res}, y_{res}$, we have $P_{(z_{res},x_{res},y_{res})} \subseteq P_{(0,0,0)}$.*

Proof. We use the following simple but important observation. Let $(u_n)_{n\in\mathbb{N}}$ be an LRS where all roots have modulus 1, i.e., each root is of the form $\gamma = e^{\mathrm{i}\theta}$, with distinct values of θ. Let u_j be the j^{th} element of the LRS, with $j \in \mathbb{N}$. Then for all ε, N, there exists $n > N$ with $|u_n - u_j| < \varepsilon$. That is, for each value visited, the LRS will visit arbitrarily close values an infinite number of times. This is the case in particular of v_n^{dom}.

Now, assume for contradiction that there is a configuration $(z_{dom}, x_{dom}, y_{dom})$ in $P_{(z_{res},x_{res},y_{res})} \setminus P_{(0,0,0)}$. Since $(z_{dom}, x_{dom}, y_{dom}) \notin P_{(0,0,0)}$, there exists m with $v_m^{dom}(z_{dom}, x_{dom}, y_{dom}) < 0$. We let $\varepsilon = \frac{|v_m^{dom}(z_{dom},x_{dom},y_{dom})|}{3}$ and N such that for all $n > N$, $|v_n^{res}| < \varepsilon$ (because it converges towards 0 when n tends towards infinity). From the above observation, we obtain an $n > N$ such that $|v_n^{dom}(z_{dom}, x_{dom}, y_{dom}) - v_m^{dom}(z_{dom}, x_{dom}, y_{dom})| < \varepsilon$. Thus:

$$\begin{aligned} v_n(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res}) &< v_n^{dom}(z_{dom}, x_{dom}, y_{dom}) + v_n^{res}(z_{res}, x_{res}, y_{res}) \\ &< v_m^{dom}(z_{dom}, x_{dom}, y_{dom}) + \varepsilon + \varepsilon \quad < 0. \end{aligned}$$

A contradiction with $(z_{res}, x_{dom}, y_{dom}) \in P_{(z_{res},x_{res},y_{res})}$. ◀

5 Proof of Theorem 8

5.1 Intuition for hardness of (robust) positivity

Consider a vector $\mathbf{d} = (z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res})$ on the surface of P_{dom}, that is, $(z_{dom}, x_{dom}, y_{dom}) \in P_{(0,0,0)}$. Consider the subset of $P_{(0,0,0)}$ which consists of points whose first coordinate z_{dom} is the same as that of $\mathbf{d}$. For all n, let $\mathbf{e}_n$ be the point of this section where hyperplane H_n is tangent to $P_{(0,0,0)}$. Let τ be the angle made between the center b of the section, $\mathbf{e}_0$ and $\mathbf{d}$. Hence, $\mathbf{e}_0$ is at angle 0 and $\mathbf{e}_n$ at angle $2\pi n\theta \mod 2\pi$. We depict this pictorially in Figure 3.

We have that $u_n(\mathbf{d}) \geq 0$ for all n iff $\mathbf{d}$ is in the intersection of all half spaces defined by $H_i(z_{res}, x_{res}, y_{res})$. As $2\pi n\theta \mod 2\pi$ is dense in $[0, 2\pi)$, for all $\beta > 0$, there is a n such that $\mathbf{e}_n$ is at angle $\alpha_n \in [\tau - \beta, \tau + \beta]$, hence H_n will be ε-close to $\mathbf{d}$. To know whether $\mathbf{d}$ is in the half space defined by $H_n(z_{res}, x_{res}, y_{res})$, we need to compare the distance ε between H_n and $\mathbf{d}$, with the value of n. If the value of n is too large, then the distance between $H_n(z_{res}, x_{res}, y_{res})$ and H_n is smaller than ε, and $\mathbf{d}$ is in the half space $H_n^+(z_{res}, x_{res}, y_{res})$.

In other words, for $(u_n(\mathbf{d}))_{n\in\mathbb{N}}$ not to be positive, n needs to be both small enough and such that $2\pi n\theta \mod 2\pi$ is close to τ. This is similar to $L(\theta)$ being small, as shown in Lemma 13.

Now, for robust positivity (Theorem 8), we consider a ball $\mathcal{B}$ entirely in P_{dom}, tangent to the surface of P_{dom} only on point $\mathbf{d}$. The ball will be positive iff the curvature of the ball is steeper than the curvature from hyperplanes $H_n(z_{res}, x_{res}, y_{res})_{n\in\mathbb{N}}$ around $\mathbf{d}$, as shown in Lemma 14. This will correspond again to computing $L(\theta)$, thus showing hardness.

5.2 Formalizing the proof for closed balls and robust positivity

In this section, we formalize the intuition given above, in the case of a closed ball and for robust positivity. We will extend this to the full proof of Theorem 8 in the next subsection.

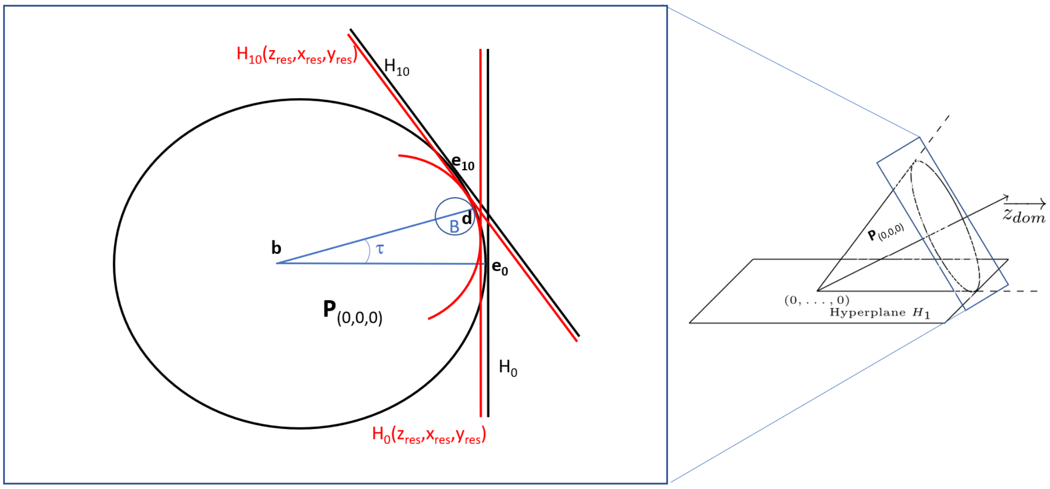


Figure 3 Representation of a section of $P_{(0,0,0)}$, with hyperplanes H_0, H_{10} being represented.

We start by picking $L(\theta) = \inf(c \in \mathbb{R} \mid |\theta - \frac{k}{n}| \leq \frac{c}{n^2}, k, n \in \mathbb{N} \setminus \{0\})$, i.e., $L(\theta) = \inf(c \in \mathbb{R} \mid |2\pi n\theta - 2\pi k| \leq \frac{2\pi c}{n}, k, n \in \mathbb{N} \setminus \{0\})$. Denoting $L^+(\theta) = \inf(c \in \mathbb{R} \mid 2\pi n\theta \mod 2\pi \leq \frac{2\pi c}{n}, n \in \mathbb{N})$ and $L^-(\theta) = \inf(c \in \mathbb{R} \mid | - 2\pi n\theta \mod 2\pi| \leq \frac{2\pi c}{n}, n \in \mathbb{N})$, we get $L(\theta) = min(L^+(\theta), L^-(\theta))$.

We show how to ε-approximate $L^+(\theta)$ in the following, using an oracle for robust positivity, following ideas in [24]. To compute some ℓ that is ε-close to $L^+(\theta)$ for a given $\varepsilon > 0$, we perform a binary search on ℓ. An old observation of Dirichlet shows that every real number has Diophantine approximation type at most 1. Further, $L(\theta) \geq 0$ by definition. So, for the binary search, we start with a lower bound $\ell_{min} = 0$ and an upper bound $\ell_{max} = 1$. For $\ell := \frac{\ell_{min}+\ell_{max}}{2}$, we want to know if $\ell \geq L^+(\theta) - \varepsilon$ (and then we set $\ell_{min} := \ell$) or whether $\ell \leq L^+(\theta) + \varepsilon$ (and then we set $\ell_{max} := \ell$). Approximating $L^-(\theta)$ is done in a symmetric way, and $L(\theta)$ can be approximated accordingly.

For an interval I of $\mathbb{N}$, we denote $L^+_I(\theta) = \inf(c \in \mathbb{R} \mid 2\pi n\theta \mod 2\pi \leq \frac{2\pi c}{n}, n \in I)$. For instance, we have $L^+_{\mathbb{N}}(\theta) = L^+(\theta)$. We will denote $> n_1$ for the interval $I = \{n_1+1, n_1+2, \ldots\}$.

Let $\varepsilon > 0$ and ℓ be a guess to check against $L^+(\theta)$. Consider the closed ball $\mathcal{B}^\ell_\psi$ of radius $\sqrt{2}\psi$, centered at $\mathbf{c} = (2+\psi, 2-\psi, 0, 0, 0, 2\pi\ell)$, with $\psi < \frac{1}{3}$ and $\psi < \pi\ell$. Notice that $\mathbf{d} = (2, 2, 0, 0, 0, 2\pi\ell) \in \mathcal{B}^\ell_\psi$, on its surface, as $||\mathbf{c} - \mathbf{d}|| = \sqrt{2}\psi$. The ball $\mathcal{B}^\ell_\psi$ is entirely in P_{dom} (see Lemma 20 in Appendix A.3, which is not necessary for the rest of the proofs, it is a sanity check because of Lemma 12). Further, the surface of the ball is tangent to the surface of P_{dom} in $\mathbf{d}$ as $2^2 = 4 = (2+0)^2$ satisfies the equation of Lemma 10. In other words, this the only point where the ball $\mathcal{B}^\ell_\psi$ intersects the surface of P_{dom}.

We first explain the relationship between the positivity of $(u_n(\mathbf{d}))$ and $L(\theta)$, which is the crux of the proof of Theorem 6 by [24].

▶ **Lemma 13.** *There is a computable $n_1 > 0$ such that for all $n_2 \geq n_1$, we have $(u_n(\mathbf{d}))_{n>n_2}$ positive implies $L^+_{>n_2}(\theta) > \ell - \varepsilon$ and $(u_n(\mathbf{d}))_{n>n_2}$ not positive implies $L^+_{>n_2}(\theta) < \ell + \varepsilon$.*

Proof. Let $\alpha_n = 2\pi n\theta \mod 2\pi \geq 0$. Considering the Taylor development for $\alpha_n > 0$ close to 0 of $(1 - \cos(\alpha_n))$ and $\sin(\alpha_n)$, we get $u_n(\mathbf{d}) = \frac{2}{2}\alpha_n^2 - \frac{2\pi\ell\alpha_n}{n} + f(\alpha_n)$, with $f(\alpha_n) = O(\alpha_n^3)$. We have $u_n(\mathbf{d}) \leq 0$ iff $\frac{2\pi\ell\alpha_n}{n}$ is larger than $\alpha_n^2(1 + \frac{f(\alpha_n)}{\alpha_n^2})$, that is iff $\alpha_n \leq \frac{2\pi\ell}{n(1+\frac{f(\alpha_n)}{\alpha_n^2})}$.

There exists a value $\alpha_0 > 0$ such that $\alpha_n < \alpha_0$ implies $1 - \frac{\varepsilon}{\ell} \leq \frac{1}{(1+\frac{f(\alpha_n)}{\alpha_n^2})} \leq 1 + \frac{\varepsilon}{\ell}$. That is, if $u_n(\mathbf{d}) \leq 0$ and $\alpha_n < \alpha_0$, then $L^+(\theta) \leq \ell + \varepsilon$. Let $n_0 = \lfloor \frac{\pi\ell}{1-\cos(\alpha_0)} \rfloor + 1$. As $|\sin(\alpha)| \leq 1$, if $\alpha_n > \alpha_0$, then for all $n > n_0$, $u_n(\mathbf{d}) > 2(1-\cos(\alpha_0)) - 2\pi\ell\frac{1-\cos(\alpha_0)}{\pi\ell} = 0$ is positive. We define $n_1 = \max(n_0, \lfloor \frac{2\pi(\ell-\varepsilon)}{\alpha_0} \rfloor + 1)$.

That is, if $u_n(\mathbf{d}) \leq 0$ with $n > n_1$, then $n > n_0$ and $\alpha_n < \alpha_0$, and thus $L^+_{>n_1}(\theta) \leq \ell + \varepsilon$.

Otherwise, for all $n > n_1$, we have $u_n(\mathbf{d})$ is positive and $2\pi n\theta \mod 2\pi > \ell - \varepsilon$. Thus we have $L^+_{>n_1}(\theta) \geq \ell - \varepsilon$. ◀

The ball $\mathcal{B}^\ell_\psi$ is chosen to have the following crucial Lemma to approximate $L^+(\theta)$:

▶ **Lemma 14.** *If $L^+(\theta) \geq \ell + \varepsilon$, there exists an explicitly computable ψ such that $u_n(\mathbf{d}') \geq 0$ for all $n > n_1$ and all $d' \in \mathcal{B}^\ell_\psi$, for the n_1 from Lemma 13.*

The proof of Lemma 14 uses Lemmas 10, 11 and the description of $H_n(z_{res}, x_{res}, y_{res})$ as parallel and at a bounded distance to H_n.

Proof. Let $\mathbf{e} = (2+\psi+z'_{dom}, 2-\psi+x'_{dom}, y_{dom}, z_{res}, x_{res}, 2\pi\ell + y'_{res}) \in \mathcal{B}^\ell_\psi$, and use the same notation $\lambda_1, \lambda_2, \lambda_3$ as in the proof of Lemma 20. We write $\lambda_3 = \cos(\beta)$, and we get $x'_{dom} = \sqrt{2}\cos(\beta)\lambda_2\lambda_1\psi$ and $y'_{dom} = \sqrt{2}\sin(\beta)\lambda_2\lambda_1\psi$.

Consider the Circle C_{dom}, section of $P_{(0,0,0)}$ over $\overrightarrow{x_{dom}}, \overrightarrow{y_{dom}}$ for $z_{dom} = 2+\psi+z'_{dom}$. It is of diameter $2+\psi+z'_{dom}$. Let α the angle $(\mathbf{bd'}, \mathbf{be})$ with $\mathbf{b} = (2+\psi+z'_{dom}, 0, 0, z_{res}, x_{res}, 2\pi\ell + y'_{res})$ and $\mathbf{d'} = (2+\psi+z'_{dom}, 2+\psi+z'_{dom}, 0, z_{res}, x_{res}, 2\pi\ell + y'_{res})$.

Consider r the distance between $\mathbf{b}$ and $\mathbf{e}$. We have $\cos(\alpha) = \frac{2-(1-\sqrt{2}\cos\beta|\lambda_2|\lambda_1)\psi}{r}$. Hence $x \geq 2 - \psi \geq 1$. We also have $\sin\alpha = \frac{\sqrt{2}|\lambda_2|\lambda_1\psi\sin\beta}{r} \leq \psi$. Thus α is small wrt 1, and $r = \frac{2-(1-\sqrt{2}\cos\beta|\lambda_2|\lambda_1)\psi}{\cos(\alpha)} = (1+\mathcal{O}(\alpha^2))(2-(1-\sqrt{2}\cos\beta|\lambda_2|\lambda_1)\psi)$.

We want to know whether $\mathbf{e}$ is in $P_{(z_{res}, x_{res}, 2\pi\ell+y'_{res})}$. It is not the case iff there exists an half space $H^+_n(z_{res}, x_{res}, 2\pi\ell + y'_{res})$ such that $\mathbf{e} \notin H^+_n(z_{res}, x_{res}, 2\pi\ell + y'_{res})$. Take n with $n\theta \mod 2\pi < \alpha$. As $L^+(\theta) \geq \ell + \varepsilon$, we have $n > \frac{2\pi(\ell+\varepsilon)}{\alpha} > \frac{2\pi\ell}{\alpha}$. That is, the remainder for this α is bounded by $\frac{\psi\sqrt{2-2\lambda_1^2}}{n} + \frac{2\pi\ell}{n}\sin\alpha < \alpha(\sin\alpha + \frac{\psi\sqrt{2-2\lambda_1^2}}{2\pi\ell})$. The diameter of the C_{dom} circle is $2+\psi+z'_{dom}$. By Lemma 10 and the description of $H_n(z_{res}, x_{res}, y_{res})$ as parallel to H_n,

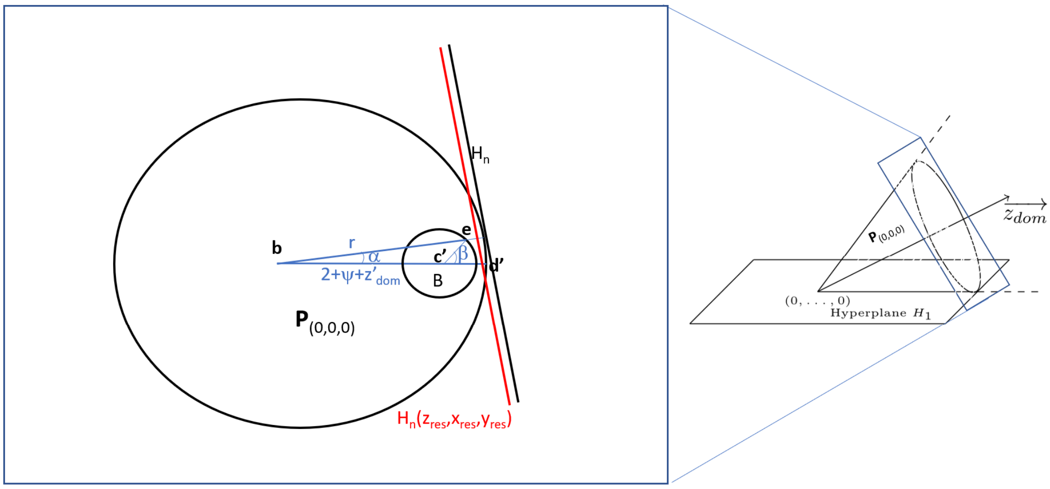


Figure 4 Representation of $\mathcal{B}$ in the section over $\overrightarrow{x_{dom}}, \overrightarrow{y_{dom}}$ at height $\ell = 2+\psi+z'_{dom}$ over $\overrightarrow{z_{dom}}$.

characterized by Lemma 11, and at a distance from H_n which we can effectively bound for all $n \in \mathbb{N}$, we obtain that if r is smaller than $2 + \psi \pm \sqrt{2}\sqrt{1 - \lambda_2^2}\lambda_1\psi - \alpha(\sin\alpha + \frac{\psi\sqrt{2-2\lambda_1^2}}{2\pi\ell})$, then $\mathbf{e}$ is in $P_{(z_{res}, x_{res}, 2\pi\ell + y'_{res})}$.

That is, we want $2+\psi\pm\sqrt{2}\sqrt{1 - \lambda_2^2}\lambda_1\psi-\alpha(\sin\alpha+\frac{\psi\sqrt{2-2\lambda_1^2}}{2\pi\ell})-r = \psi(2\pm\sqrt{2}\sqrt{1 - \lambda_2^2}\lambda_1 - \sqrt{2}\cos\beta|\lambda_2|\lambda_1) - \alpha\frac{\sqrt{2-2\lambda_1^2}}{2\pi\ell})) + \mathcal{O}(\alpha^2) > 0$. Now remark that $|\sqrt{2}\sqrt{1 - \lambda_2^2}\lambda_1\psi + \sqrt{2}\cos\beta|\lambda_2|\lambda_1| \leq \lambda_1(\sqrt{2+2\cos^2\beta})$. And also $|\lambda_1(\sqrt{2+2\cos^2\beta}) + \alpha\frac{\psi\sqrt{2-2\lambda_1^2}}{2\pi\ell})| \leq \sqrt{2 + 2\cos^2\beta + \alpha^2\frac{\psi^2}{\pi\ell}}$, applying twice the same reasoning as in the proof of Lemma 15.

We now prove that we have $\psi(1 - \cos^2\beta) = \psi\sin^2\beta$ dominates any $\mathcal{O}(\alpha^2)$ for ψ small, i.e., we prove that for any $f = \mathcal{O}(\alpha^2)$, we have $\frac{f}{\psi(\sin^2\beta)}$ tends to 0 as ψ tends to 0. In particular, for all ψ small enough, the fraction is below 1, i.e., $(\sin^2\beta)\psi > f$. We have $(\sin^2\beta)\psi \geq \frac{r^2\sin^2(\alpha)}{\psi} \geq \frac{\sin^2(\alpha)}{\psi}$ as $r \in [1,3]$. This indeed dominates any function $\mathcal{O}(\alpha^2)$, as $\frac{\alpha}{\sin\alpha}$ is bounded in $[-\frac{\pi}{2}, \frac{\pi}{2}]$. In particular, $\sin^2\beta > \alpha^2\frac{\psi^2}{2\pi\ell})$ for ψ small enough.

Now, we write $\sqrt{2 + 2\cos^2\beta + \alpha^2\frac{\psi^2}{\pi\ell}} = 2\sqrt{1 - \frac{1}{2}(\sin^2\beta - \alpha^2\frac{\psi^2}{2\pi\ell})} \leq 2(1 - \frac{1}{4}(\sin^2\beta - \frac{1}{4}\alpha^2\frac{\psi^2}{2\pi\ell}))$ using the Taylor development of $\sqrt{1-r}$, and the fact that $(\sin^2\beta - \alpha^2\frac{\psi^2}{2\pi\ell}) > 0$ because ψ is small enough. Thus, we obtain $\psi(2-2(1-\frac{1}{4}(\sin^2\beta))+\mathcal{O}(\alpha^2) = \frac{\psi}{4}(\sin^2\beta)+\mathcal{O}(\alpha^2)$, which is positive for ψ small enough, and $\mathbf{e}$ is in $P_{(z_{res}, x_{res}, 2\pi\ell + y'_{res})}$.

Notice that the function in $\mathcal{O}(\alpha^2)$ is well defined and well known, and thus Ψ small enough can be effectively computed. ◀

Let us explain why these two Lemmas suffice, provided that we have an oracle for ψ-robust positivity, to answer either $L^+(\theta) \leq \ell + \varepsilon$ or $L^+(\theta) \geq \ell - \varepsilon$, which proves Theorem 8 for robust positivity and closed balls. Intuitively, if the ball $\mathcal{B}^\ell_\psi$ is positive, then in particular $(u_n(\mathbf{d}))$ is positive since $\mathbf{d} \in \mathcal{B}^\ell_\psi$ and we have $L^+_{>n_1}(\theta) > \ell - \varepsilon$ by Lemma 13. Otherwise, the ball is not positive and Lemma 14 shows that $L^+(\theta) < \ell + \varepsilon$, granted that the radius of the ball is small enough.

Proof of Theorem 8 for robust positivity and closed balls. Let $\varepsilon > 0$. Assume that an ℓ has been fixed, such that we want to know either $L^+(\theta) < \ell + \varepsilon$ or $L^+(\theta) > \ell - \varepsilon$. First, we fix ψ given by Lemma 14. We remark that $\mathcal{B}^\ell_\psi$ corresponds to a ball in the coordinates $(z_{dom}, x_{dom}, y_{dom}, z_{res}, x_{res}, y_{res})$ (which are not necessarily orthonormal), not in the original coordinates $(v_0, v_1, v_2, v_3, v_4, v_5)$. Taking the transformation from the latter to the former, which is a linear operator H, the ball $\mathcal{B}^\ell_\psi$ corresponds to an ovaloid $\mathcal{O}$ in the original coordinates. We can explicitly define a ball $\mathcal{B}' \subseteq \mathcal{O}$ in the original coordinates, with $\mathbf{d} \in \mathcal{B}'$. Notice that we can choose $\mathcal{B}'$ with an arbitrarily small radius, so in particular we can choose this radius to be rational without loss of generality.

We first compute $L^+_{\leq n_1}(\theta)$, which is easy as it only involves a bounded number of indices n. If $L^+_{\leq n_1}(\theta) < \ell + \varepsilon$, then we know $L^+(\theta) \leq L^+_{\leq n_1}(\theta) < \ell + \varepsilon$ and we stop.

Otherwise $L^+_{\leq n_1}(\theta) \geq \ell + \varepsilon$, and we check whether $u_n(\mathbf{d}') \geq 0$ for all $n > n_1$ and all $d' \in \mathcal{B}'$, using the robust positivity oracle (by starting from $\mathbf{M}^{n_1}(v_0, \dots, v_k)$ rather than $(v_0, \dots, v_k)$). If it is positive, then in particular it is for $\mathbf{d}' = \mathbf{d}$, and applying Lemma 13, we obtain $L^+_{>n_1}(\theta) > \ell - \varepsilon$. Combined with $L^+_{\leq n_1}(\theta) \geq \ell + \varepsilon$, we obtain $L^+(\theta) > \ell - \varepsilon$.

The last case means that there is $u_n(\mathbf{d}') < 0$ for some $n > n_1$ and $d \in B' \subseteq B^\ell_\psi$. Applying the contrapositive of Lemma 14, we obtain that $L^+(\theta) < \ell + \varepsilon$. ◀

5.3 Case of Open Balls and robust Skolem

In this subsection, we extend the proof of Theorem 8 to show that considering open or closed balls does not make a difference for the Diophantine-hardness. Further, there is also no difference whether we consider the robust Skolem problem (0 is avoided), the robust positivity problem (negative numbers are avoided), or the robust strict positivity problem (negative and 0 are avoided).

Let B be an open ball and $cl(B)$ its topological closure, which is the closed ball consisting of B and its surface. Consider the following statements:

1. Robust Positivity holds for the closed ball $cl(B)$
2. Robust Positivity holds for the open ball B
3. Robust Strict Positivity holds for the open ball B
4. Robust Skolem holds for the open ball B
5. Robust Strict Positivity holds for the closed ball $cl(B)$
6. Robust Skolem holds for the closed ball $cl(B)$

We show that equivalence results between these statements. This allows us to conclude that having open or closed balls does not make a difference for $\mathcal{T}$-Diophantine hardness of Skolem and (strict) positivity. Formally, we have the following.

▶ **Lemma 15.** *(1), (2) and (3) are equivalent. Further, for balls B containing at least one initial configuration $\mathbf{d}_0$ in its interior that is strictly positive, i.e. $u_n(\mathbf{d}_0) > 0$ for all n, both (3) and (4) are equivalent and (5) and (6) are equivalent.*

Proof. (1) implying (2) is trivial. (2) implies (1): we show the contrapositive. Suppose there exists an initial configuration $\mathbf{d}$ on the surface of the ball B and an integer n such that $u_n(\mathbf{d}) = y < 0$. Recall that M is the companion matrix, and $u_n(\mathbf{d})$ is the first component of $(M^n.d)$, so $u_n(x)$ is a continuous function. Thus, there exists a neighbourhood of $\mathbf{d}$, such that for all $\mathbf{d}'$ in the neighbourhood, $u_n(\mathbf{d}') < y/2 < 0$. This neighbourhood intersects the open ball B enclosed by the surface, and picking d' in this intersection shows that Robust Positivity does not hold in the open ball.

(3) implying (2) is trivial. (2) implies (3): Assume for the sake of contradiction that there is an initial configuration $\mathbf{c}'$ in the open ball B such that $u_n(\mathbf{c}') = 0$. Consider any open O around $\mathbf{c}'$ entirely in the open ball B. We have that $\mathbf{c}'$ is on hyperplane H_n by definition. That is, there are initial configurations in O on both sides of H_n. In particular, there is an initial configuration $\mathbf{c}''$ in O, hence in B, with $\mathbf{c}'' \notin H_n^+$, i.e. $u_n(\mathbf{c}'') < 0$, a contradiction with B being robustly positive.

(3) implies (4) is trivial. (4) implies (3): We consider the contrapositive: if we have an initial configuration $\mathbf{d}_1$ of B which is not strictly positive, then $u_n(\mathbf{d}_1) \leq 0$ for some n, and there is a barycenter $\mathbf{d}_2$ between $\mathbf{d}_0, \mathbf{d}_1$ which satisfies $u_n(\mathbf{d}_2) = 0$, i.e. negation of (4). To be more precise, we can choose $\mathbf{d}_2 = \frac{-u_n(\mathbf{d}_1)}{u_n(\mathbf{d}_1)-u_n(\mathbf{c})}\mathbf{d}_0 + \frac{u_n(\mathbf{d}_1)}{u_n(\mathbf{d}_1)-u_n(\mathbf{d}_0)}\mathbf{d}_1$.

Now, (5) and (6) are equivalent for balls containing at least one initial configuration $\mathbf{d}_0$ that is strictly positive in its interior (same proof as for the equivalence between (3) and (4) above). However, notice that (5,6) are not equivalent with (1,2,3,4) in general. ◀

We are now ready to prove Theorem 8 for open balls B. It suffices to remark that the center $\mathbf{c}$ of B_ψ^ℓ is strictly in the interior of P_{dom}, and thus it will be eventually strictly positive by Lemma 10, that is there exists $n_2 > n_1$ such that $u_n(\mathbf{c}) > 0$ for all $n > n_2$, and we can choose $\mathbf{d}_0 = \mathbf{c}$. Hence by Lemma 15, robustness (for $n > \max(n_1, n_2)$) of positivity, strict positivity and Skolem are equivalent on $\mathcal{B}$, and these are equivalent with robust positivity of $cl(\mathcal{B})$ which was proved $\mathcal{T}$-Diophantine hard in the previous section.

It remains to prove Theorem 8 for robust Skolem for closed balls $\mathcal{B}$. For that, it suffices to easily adapt Lemma 13, replacing $(u_n(\mathbf{d}))_{n>n_2}$ positive by strictly positive, and obtain the $\mathcal{T}$-Diophantine hardness for robust strict positivity of closed balls. We again apply Lemma 15 ((5) and (6) are equivalent) to obtain hardness for robust Skolem of closed balls.

6 Proof of Theorem 9

We now turn to the proof of Theorem 9, generalizing elements from Section 4.

6.1 Intuitions for the proof of Theorem 9

Let $(u_n)_{n\in\mathbb{N}}$ be a recurrence relation defined by coefficients $\mathbf{a} \in \mathbb{Q}^\kappa$. As before, we will consider $(v_n)_{n\in\mathbb{N}} = (\frac{u_n}{f_n})_{n\in\mathbb{N}}$, for f_n such that the dominant coefficients of $(v_n)_{n\in\mathbb{N}}$ are of the form $\alpha e^{in\theta}$. We will then decompose the exponential solution of $(v_n)_{n\in\mathbb{N}}$ as a *dominant term* $(v_n^{dom})_{n\in\mathbb{N}}$ made of coefficients $\alpha e^{in\theta}$, and a *residue* $(v_n^{res})_{n\in\mathbb{N}}$ with $(v_n^{res})_{n\in\mathbb{N}} \underset{n\to+\infty}{\longrightarrow} 0$. For an initial distribution $\mathbf{a}$, we denote by $\mathbf{a}^{dom}$ its projection on dominant space. As before, we define $P_{dom} = \{\mathbf{a} \mid \forall n, v_n^{dom}(\mathbf{a}^{dom}) \geq 0\}$.

To solve $\exists$-robust Skolem and $\exists$-robust positivity, the reasoning is based on the range of the dominant term. For $\exists$-robust Skolem, we consider the minimum *absolute* value ν of the dominant term $|v_n^{dom}(\mathbf{c}_0)|$ obtained for the center of the neighborhood $\mathbf{c}_0$.

- $\nu > 0$. Then as the residue has negligible contribution to $(v_n)_{n\in\mathbb{N}}$ for large n, we show that the LRS will ultimately avoid zero beyond a threshold index n_{thr}. Having assured ourselves of the long run behaviour, it suffices to check the value of the LRS up to n_{thr}, where the residue can have significant contribution, to see whether the LRS satisfies robust Skolem.
- $\nu = 0$. Then we show in Proposition 18 that the LRS does not satisfy robust Skolem: no matter how small we pick a neighbourhood around $\mathbf{c}_0$, there will always exist a $\mathbf{c}$ in that neighbourhood that hits zero at some iteration.
- Further, Proposition 17 states that ν can be computed effectively.

For robust positivity, we let μ be the minimum value of the dominant term (and not of its absolute value). Thus, μ can take three kinds of values: $\mu > 0$ ($\mathbf{c}_0 \in P_{dom}$) and we proceed as for $\nu > 0$; $\mu < 0$ ($\mathbf{c}_0 \notin P_{dom}$) and then there exists a n such that the LRS from $\mathbf{c}_0$ is negative; and $\mu = 0$ ($\mathbf{c}_0$ is at the surface of P_{dom}), and then we can show that there exists a configuration arbitrarily close to $\mathbf{c}$ such that the LRS from that configuration is negative.

6.2 Range of the Dominant Term

We first define the normalized exponential polynomial solution $(v_n)_{n\in\mathbb{N}}$:

▶ **Definition 16.** *Let $(u_n)_{n\in\mathbb{N}}$ be an LRS of general term $u_n(\mathbf{c}) = \sum_{i=1}^{r}\sum_{j=0}^{m_r-1} p_{ij} n^j \gamma_i^n$, with ρ being the modulus of the dominant roots and $m+1$ the maximal multiplicity of a dominant root. Define $v_n(\mathbf{c}) = \frac{u_n(\mathbf{c})}{n^m \rho^n}$ for $n > 0$, and $v_0(\mathbf{c}) = u_0(\mathbf{c})$.*

We call every term of v_n which converges towards 0 as n tends towards infinity residual, while the other terms, of the form $\alpha e^{i\theta}$ are dominant. We denote $\{\theta_j \mid j = 1, \ldots, k\}$ the set of θ in dominant terms, and $\alpha_j(\mathbf{c})$ the associated coefficient. We define:

$$v_n^{dom}(\mathbf{c}) = \sum \alpha_j(\mathbf{c}) e^{in\theta_j} \quad \text{and} \quad v_n^{res}(\mathbf{c}) = v_n(\mathbf{c}) - v_n^{dom}(\mathbf{c}) = \mathcal{O}(\frac{1}{n}) \rightarrow_{n\to\infty} 0.$$

As we explained in Section 6.1, knowing the range of $(v_n^{dom})_{n\in\mathbb{N}}$ is crucial in order to solve $\exists$-robust Skolem and positivity. In Section 4 and 5, we dealt with hardness via an example which had 3 dominant roots, and it was rather simple to determine the min/max value (computed in the proof of Lemma 10). The general case is not so easy however, because there may be relationships between the θ_j which may alter the range of $(v_n^{dom})_{n\in\mathbb{N}}$.

A tedious but now rather classical way to compute the range of $(v_n^{dom})_{n\in\mathbb{N}}$ is by invoking Masser's theorem [19] (Theorem 21 in Appendix A.4), to describe the set of tuples that can be reached (at least arbitrarily close) by $\mathbf{s}^n = (e^{in\theta_1}, \dots, e^{in\theta_k})$ for $n \in \mathbb{N}$, as used in [23, Theorem 4]. We can describe a continuous relaxation of $\{\mathbf{s}^n \mid n \in \mathbb{N}\}$ as a set T of tuples $\mathbf{t} = (t_1, \dots, t_k)$ of complex numbers with $|t_j| = 1$ for all j. The set T is the set of linear combinations of the finite basis given by Theorem 21, which describes a Torus, *independent of the initial configuration* $\mathbf{c}$. Notice that T may be discrete and have finitely many points (the case where $\frac{\theta_j}{2\pi} \in \mathbb{Q}$), else, it is *continuous* and has uncountably many points.

We have $\mathbf{s}^n \in T$ for all $n \in \mathbb{N}$. Further, Kronecker [11] (Theorem 22 in Appendix A.4) implies that for all $\mathbf{t} = (t_1, \dots, t_k) \in T$ and $\varepsilon > 0$, there exists an n such that $|t_j - e^{in\theta_j}| \leq \varepsilon$ for all $j \leq k$. For every initial configuration $\mathbf{c}$ and element of the torus $\mathbf{t} \in T$, we denote $\text{dominant}(\mathbf{c}, \mathbf{t}) = \sum_j \alpha_j(\mathbf{c}) t_j$. Thus, for all n and all $\mathbf{c}$, we have $v_n^{dom}(\mathbf{c}) = \text{dominant}(\mathbf{c}, \mathbf{s}^n)$. Conversely, for all $\mathbf{t} \in T, \varepsilon > 0$, there exists n with $|v_n^{dom}(\mathbf{c}) - \text{dominant}(\mathbf{c}, \mathbf{t})| \leq \varepsilon$, for all $\mathbf{c}$.

Using Renegar's result [26], one can compute effectively the range of $\text{dominant}(\mathbf{c}, \mathbf{t})$ over T, and thus of $(v_n^{dom})_{n\in\mathbb{N}}$. A simple adaptation allows to compute the range of $|\text{dominant}(\mathbf{c}, \mathbf{t})|$. In the following, we fix $\mathbf{c}_0$ to be the center of the neighborhood and define

$$\mu = \min_{\mathbf{t}\in T}(\text{dominant}(\mathbf{c}_0, \mathbf{t})) \quad \text{and} \quad \nu = \min_{\mathbf{t}\in T} |\text{dominant}(\mathbf{c}_0, \mathbf{t})|.$$

▶ **Proposition 17.** *μ and ν are algebraic and can be efficiently computed. Further, we have $|\mu|, |\nu| < 2^{s^{\mathcal{O}(1)}}$ and $\frac{1}{|\mu|}, \frac{1}{|\nu|} < 2^{s^{\mathcal{O}(1)}}$*

Proof. The statement for μ comes directly from Renegar [26], stating that we can compute the min and max values $\mu = \min_{\mathbf{t}\in T}(\text{dominant}(\mathbf{c}_0, \mathbf{t}))$ and $\mu' = \max_{\mathbf{t}\in T}(\text{dominant}(\mathbf{c}_0, \mathbf{t}))$ over t in the torus T. The statement for ν is a corollary obtained as follows:

- If $\mu > 0$ ($u_n(\mathbf{c}_0) \geq \mu > 0$ for all $n \in \mathbb{N}$), then $\nu = \mu$ and we are done.
- If $\mu' < 0$ ($u_n(\mathbf{c}_0) \leq \mu' < 0$ for all n), then $\nu = -\mu'$.
- If T is discrete, we enumerate the polynomially many values of $\mathbf{t}$ (noting that they all correspond to λ^{th} roots of unity, and Masser polynomially bounds λ) to compute ν as the minimum of the absolute values.
- Otherwise, we have $\mu < 0 < \mu'$, that is there exist two elements $\mathbf{t}, \mathbf{t}' \in T$ with $\text{dominant}(\mathbf{c}_0, \mathbf{t}) < 0 < \text{dominant}(\mathbf{c}_0, \mathbf{t}')$. As $\mathbf{x} \mapsto \text{dominant}(\mathbf{c}_0, \mathbf{x})$ is continuous over T, there is a $\mathbf{t}'' \in T$ with $\text{dominant}(\mathbf{c}_0, \mathbf{t}'') = 0$, that is $\nu = 0$. ◀

We now state that if $\mu \leq 0$, then $\exists$-robust positivity does not hold, while if $\nu = 0$, then $\exists$-robust Skolem does not hold.

▶ **Proposition 18.** *If $\mu \leq 0$, then $\forall \varepsilon > 0$, $\exists n, \mathbf{c}_\varepsilon$ with $|\mathbf{c}_0 - \mathbf{c}_\varepsilon| \leq \varepsilon$ such that $v_n(\mathbf{c}_\varepsilon) < 0$.*
If $\nu = 0$, then $\forall \varepsilon > 0$, $\exists n, \mathbf{c}_\varepsilon$ with $|\mathbf{c}_0 - \mathbf{c}_\varepsilon| \leq \varepsilon$ such that $v_n(\mathbf{c}_\varepsilon) = 0$.

To prove Proposition 18, we reason as follows. For every n, let *distance*$(\mathbf{c}, H_n)$ be the distance between an initial configuration $\mathbf{c}$ and the hyperplane $H_n = \{\mathbf{c}' \mid v_n(\mathbf{c}') = 0\} = \{\mathbf{c}' \mid u_n(\mathbf{c}') = 0\}$. If *distance*$(\mathbf{c}_0, H_n) < \varepsilon$, then there exists a $\mathbf{c}_\varepsilon$ with $|\mathbf{c}_0 - \mathbf{c}_\varepsilon| \leq \varepsilon$ such that $v_n(\mathbf{c}_\varepsilon) = 0$ ($\exists$-robust Skolem does not hold). It also implies the existence of a $\mathbf{c}'_\varepsilon$ with $|\mathbf{c}_0 - \mathbf{c}'_\varepsilon| \leq \varepsilon$ and $v_n(\mathbf{c}'_\varepsilon) < 0$, as there will be initial configurations in the ε neighborhood of $\mathbf{c}_0$ on both sides of H_n, thus some will be outside of $H_n^+ = \{\mathbf{c}' \mid v_n(\mathbf{c}') \geq 0\}$, thus with $v_n(\mathbf{c}_\varepsilon) < 0$ ($\exists$-robust positivity does not hold).

▶ **Lemma 19.** *There exists C such that for all n, $distance(\mathbf{c}, H_n) \leq C \cdot |v_n(\mathbf{c})|$.*

Lemma 19, proved in Appendix A.4, implies that if for all $\alpha > 0$, there exists n_α with $|v_{n_\alpha}(\mathbf{c})| < \alpha$, then there exists n with $distance(\mathbf{c}, H_n) < \varepsilon$ (choose $n = n_\alpha$ for $\alpha = \frac{\varepsilon}{2C}$).

Proof of Proposition 18. Consider the first statement. If $\mu < 0$, then there exists n with $|v^{res}_n(\mathbf{c}_0)| \leq \frac{\mu}{2}$ as $v^{res}_n(\mathbf{c}_0) \to_{n\to\infty} 0$. Thus $v_n(\mathbf{c}_0) = v^{dom}_n(\mathbf{c}_0) + v^{res}_n(\mathbf{c}_0) < \frac{\mu}{2}$. That is, $\mathbf{c}_\varepsilon = \mathbf{c}_0$ satisfies the statement. Otherwise, $\mu = 0$. We prove that for all $\alpha > 0$, we have a n_α such that $|v_{n_\alpha}(\mathbf{c}_0)| \leq \alpha$, which suffices by Lemma 19. Let $\alpha > 0$ arbitrarily small, and let N such that for all $n > N$, $|v^{res}_n(\mathbf{c}_0)| \leq \frac{\alpha}{2}$. This N exists as $v^{res}_n(\mathbf{c}_0) \to_{n\to\infty} 0$. By Kronecker, as $\mu = 0$, there exists $n_\alpha > N$ with $|v^{dom}_{n_\alpha}(\mathbf{c}_0)| < \frac{\alpha}{2}$. Thus $|v_{n_\alpha}(\mathbf{c}_0)| \leq |v^{dom}_{n_\alpha}(\mathbf{c}_0)| + |v^{res}_{n_\alpha}(\mathbf{c}_0)| \leq \alpha$.

We now prove the second statement in the same way. Assume $\nu = 0$, and let $\varepsilon > 0$. We again prove that for all $\alpha > 0$, we have a n_α such that $|v_{n_\alpha}(\mathbf{c}_0)| \leq \alpha$, which suffices by Lemma 19. Let $\alpha > 0$ arbitrarily small, and let N such that for all $n > N$, $|v^{res}_n(\mathbf{c}_0)| \leq \frac{\alpha}{2}$. This N exists as $v^{res}_n(\mathbf{c}_0) \to_{n\to\infty} 0$. By Kronecker, as $\nu = 0$, there exists $n_\alpha > N$ with $|v^{dom}_{n_\alpha}(\mathbf{c}_0)| < \frac{\alpha}{2}$. Thus $|v_{n_\alpha}(\mathbf{c}_0)| \leq |v^{dom}_{n_\alpha}(\mathbf{c}_0)| + |v^{res}_{n_\alpha}(\mathbf{c}_0)| \leq \alpha$. ◀

6.3 Decidability and complexity for ∃-robust Skolem and ∃-robust positivity

We now turn to deciding ∃-robust Skolem and positivity as stated in Theorem 9, using Proposition 18. The algorithm for ∃-robust Skolem is as follows (as detailed in Algorithm 1 in Appendix A.4). First, we compute $\nu \leftarrow \min_{\mathbf{t}\in T} |\mathrm{dominant}(\mathbf{c}, \mathbf{t})|$ using Proposition 17, for $\mathbf{c}$ the initial configuration around which we are looking for a neighborhood. If $\nu = 0$, then ∃-robust Skolem does not hold. Otherwise, we compute N such that $v^{res}_n(\mathbf{c}_0) < \frac{\nu}{2}$ for all $n > N$. Then we check if $v_n(\mathbf{c}_0) = 0$ for some $n \leq N$. If yes, then ∃-robust Skolem does not hold, otherwise it holds. This algorithm can readily be adapted to provide an $\varepsilon > 0$ such that for all $\mathbf{c}$ with $|\mathbf{c} - \mathbf{c}_0| \leq \varepsilon$, we have $u_c \neq 0$, as well as to decide robust positivity.

The correctness of the above algorithm follows from Proposition 18, because if $\nu > 0$, then for all $n > N$, $v_n(\mathbf{c}_0) > \nu - \frac{\nu}{2} \geq \frac{\nu}{2} > 0$, and this remains > 0 in a neighborhood of $\mathbf{c}_0$. Denoting $\nu'(\mathbf{c}) = min_{n\leq N}|v_n(\mathbf{c})|$, if $\nu'(\mathbf{c}_0) > 0$, then also $\nu'(\mathbf{c})$ for $\mathbf{c}$ in a neighborhood of $\mathbf{c}_0$.

We now argue about the complexity. Both μ and $1/\mu = 2^{s^{\mathcal{O}(1)}}$ are bounded (Proposition 17). We thus have $n_{thr} = 2^{s^{\mathcal{O}(1)}}$ because $v^{res}_n(\mathbf{c}_0) = \mathcal{O}(\frac{1}{n})$. This is the number of iterates we have to explicitly check, which gives the PSPACE complexity. This finally completes the proof of Theorem 9.

7 Conclusion

We have formulated a natural notion of robustness for the Skolem and positivity problems and shown several results: for a given neighborhood around an initial configuration $\mathbf{c}_0$, we show Diophantine-hardness for both problems. Interestingly, this is the first Diophantine-hardness result for a variant of Skolem as far as we know. This implies that for uninitialized positivity, the fact that the initial configuration $\mathbf{c}_0$ is arbitrary is crucial to decidability [28, 12], as having a fixed ball around $\mathbf{c}_0$ is not sufficient.

On the other hand, we proved decidability of ∃-robust Skolem/Positivity around an initial configuration in full generality, hence this problem is simpler. It is also more practical because in a real system, it is often impossible to determine the initial configuration with absolute accuracy. Our results can provide a precision with which it is sufficient to set the initial configuration. Beyond these results, we provided geometrical reinterpretations of Skolem/positivity, shedding a new light on this hard open problem.

References

1 Manindra Agrawal, S. Akshay, Blaise Genest, and P. S. Thiagarajan. Approximate verification of the symbolic dynamics of Markov chains. *J. ACM*, 62(1):2:1–2:34, 2015.

2 S. Akshay, Nikhil Balaji, Aniket Murhekar, Rohith Varma, and Nikhil Vyas. Near-Optimal Complexity Bounds for Fragments of the Skolem Problem. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154, pages 37:1–37:18, 2020.

3 S. Akshay, Nikhil Balaji, and Nikhil Vyas. Complexity of restricted variants of skolem and related problems. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 78:1–78:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

4 S. Akshay, Blaise Genest, Bruno Karelovic, and Nikhil Vyas. On regularity of unary probabilistic automata. In *STACS'16*, pages 8:1–8:14. LIPIcs, 2016.

5 S. Akshay, Blaise Genest, and Nikhil Vyas. Distribution-based objectives for Markov Decision Processes. In *33rf Symposium on Logic in Computer Science (LICS 2018)*, volume IEEE, pages 36–45, 2018.

6 Christel Baier, Florian Funke, Simon Jantsch, Toghrul Karimov, Engel Lefaucheux, Joël Ouaknine, Amaury Pouly, David Purser, and Markus A. Whiteland. Reachability in dynamical systems with rounding. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020*, volume 182 of *LIPIcs*, pages 36:1–36:17, 2020.

7 S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry.* Springer, 2nd edition, 2006.

8 Jean Berstel and Maurice Mignotte. Deux propriétés décidables des suites récurrentes linéaires. *Bulletin de la Société Mathématique de France*, 104:175–184, 1976.

9 Manuel Biscaia, David Henriques, and Paulo Mateus. Decidability of approximate Skolem problem and applications to logical verification of dynamical properties of markov chains. *ACM Trans. Comput. Logic*, 16(1), December 2014.

10 Vincent Blondel and Natacha Portier. The presence of a zero in an integer linear recurrent sequence is NP-hard to decide. *Linear algebra and its Applications*, 351:91–98, 2002.

11 N. Bourbaki. *Elements of Mathematics: General Topology (Part 2).* Addison-Wesley, 1966.

12 Mark Braverman. Termination of integer linear programs. In *International Conference on Computer Aided Verification*, pages 372–385. Springer, 2006.

13 Ventsislav Chonev, Joël Ouaknine, and James Worrell. On the Skolem problem for continuous linear dynamical systems. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pages 100:1–100:13. LIPIcs, 2016.

14 H. Cohen. *A Course in Computational Algebraic Number Theory.* Springer-Verlag, 1993.

15 Julian D'Costa, Toghrul Karimov, Rupak Majumdar, Joël Ouaknine, Mahmoud Salamati, Sadegh Soudjani, and James Worrell. The pseudo-Skolem problem is decidable. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021*, volume 202 of *LIPIcs*, pages 34:1–34:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

16 Graham Everest, Alfred J. van der Poorten, Igor E. Shparlinski, and Thomas Ward. Recurrence Sequences. In *Mathematical surveys and monographs*, 2003.

17 Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's problem - on the border between decidability and undecidability. Technical Report 683, Turku Centre for Computer Science, 2005.

18 Kurt Mahler. *Eine arithmetische Eigenschaft der Taylor-koeffizienten rationaler Funktionen.* Noord-Hollandsche Uitgevers Mij, 1935.

19 David W. Masser. Linear relations on algebraic groups. In *New Advances in Transcendence Theory.* Cambridge University Press, 1988.

20 M. Mignotte. Some useful bounds. In *Computer Algebra*, 1982.

21 Maurice Mignotte, Tarlok Nath Shorey, and Robert Tijdeman. The distance between terms of an algebraic recurrence sequence. *Journal für die reine und angewandte Mathematik*, 349:63–76, 1984.

22 Eike Neumann. Decision problems for linear recurrences involving arbitrary real numbers. *Logical Methods in Computer Science*, 17(3), 2021.

23 Joël Ouaknine and James Worrell. On the positivity problem for simple linear recurrence sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 318–329. Springer, 2014.

24 Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 366–379. SIAM, 2014.

25 Joël Ouaknine and James Worrell. Ultimate positivity is decidable for simple linear recurrence sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 330–341. Springer, 2014.

26 James Renegar. On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Part I: Introduction. Preliminaries. The Geometry of Semi-Algebraic Sets. The Decision Problem for the Existential Theory of the Reals. *J. Symb. Comput.*, 13:255–300, 1992.

27 Thoralf Skolem. Ein Verfahren zur Behandlung gewisser exponentialer Gleichungen, 8de Skand. mat. *Kongr. Forh. Stockholm*, 1934.

28 Ashish Tiwari. Termination of linear programs. In *Computer-Aided Verification, CAV*, volume 3114 of *LNCS*, pages 70–82. Springer, July 2004.

29 Nikolai Vereshchagin. The problem of appearance of a zero in a linear recurrence sequence. *Mat. Zametki*, 38(2):609–615, 1985.

A Appendix

A.1 Regarding algebraic numbers and their bit representation

A complex number α is said to be algebraic if it is a root of a polynomial with integer coefficients. For an algebraic number α, its defining polynomial p_α is the unique polynomial of least degree of $\mathbb{Z}[X]$ such that the GCD of its coefficients is 1 and α is one of its roots. Given a polynomial $p \in \mathbb{Z}[X]$, we denote the length of its representation size(p), its height $H(p)$ the maximum absolute value of the coefficients of p and $d(p)$ the degree of p. When the context is clear, we will only use H and d.

A separation bound provided in [20] has established that for distinct roots α and β of a polynomial $p \in \mathbb{Z}[x]$, $|\alpha - \beta| > \frac{\sqrt{6}}{d^{(d+1)/2}H^{d-1}}$. This bound allows one to represent an algebraic number α as a 4-tuple (p, a, b, r) where α is the only root of p at distance $\leq r$ if $a + ib$, and we denote sizeα the size of this representation, i.e., number of bits needed to write down this 4-tuple.

Further, we note that two distinct algebraic numbers α and β, are always roots of $p_\alpha p_\beta$, and we have that

$$\frac{1}{|\alpha - \beta|} = 2^{(||\alpha|| + ||\beta||)^{\mathcal{O}(1)}}. \tag{1}$$

Given a polynomial $p \in \mathbb{Z}[X]$, one can compute its roots in polynomial time wrt size(p) [7]. Since algebraic numbers form a field, given α, β two algebraic numbers, one can always compute the representations of $\alpha + \beta$, $\alpha\beta$, $\frac{1}{\alpha}$, $Re(\alpha)$, $Im(\alpha)$ in polynomial time wrt size(α) + size(β) [7, 14].

A.2 Proofs for Section 4

▶ **Lemma 11.** $\mathbf{M}_{dom}(z_{dom}, x_{dom}, y_{dom}) = (z_{dom}, x_{dom}\cos(2\pi\theta) + y_{dom}\sin(2\pi\theta), y_{dom}\cos(2\pi\theta) - x_{dom}\sin(2\pi\theta))$*, that is* $\mathbf{M}_{dom}$ *is a rotation around axis* $\overrightarrow{z}$ *of angle* $-2\pi\theta$*.*

Proof. We use the formulas $\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$ and $\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$.

Matrix $\mathbf{M}_{dom}$ transforms $v_n^{dom}(z_{dom}, x_{dom}, y_{dom})$ into $v_{n+1}^{dom}(z_{dom}, x_{dom}, y_{dom})$. Using the formulas above with $a = 2\pi n\theta, b = 2\pi\theta$, we have that for all $n \geq 1$, $v_{n+1}^{dom}(z_{dom}, x_{dom}, y_{dom}) = v_n^{dom}(z_{dom}, x_{dom}\cos(2\pi\theta) + y_{dom}\sin(2\pi\theta), y_{dom}\cos(2\pi\theta) - x_{dom}\sin(2\pi\theta))$ for all n, and thus $\mathbf{M}_{dom}$ transforms $(z_{dom}, x_{dom}, y_{dom})$ into $(z_{dom}, x_{dom}\cos(2\pi\theta) + y_{dom}\sin(2\pi\theta), y_{dom}\cos(2\pi\theta) - x_{dom}\sin(2\pi\theta))$.

Now, consider a point p in 2D space at cartesian coordinates (x_{dom}, y_{dom}). Its polar coordinates are (r, α), with $r = \sqrt{x_{dom}^2 + y_{dom}^2}$ the distance between $(0,0)$ and p. Consider the point at polar coordinates $(r, \alpha - 2\pi\theta)$. Thus it is at cartesian coordinates $(r\cos(\alpha - 2\pi\theta), r\sin(\alpha - 2\pi\theta)) = (r\cos(\alpha)\cos(2\pi\theta) + r\sin(\alpha)\sin(2\pi\theta), r\sin(\alpha)\cos(2\pi\theta) - r\cos(\alpha)\sin(2\pi\theta)) = x_{dom}\cos(2\pi\theta) + y_{dom}\sin(2\pi\theta), y_{dom}\cos(2\pi\theta) - x_{dom}\sin(2\pi\theta))$. Hence the rotation of angle $-2\pi\theta$ transforms (x_{dom}, y_{dom}) into $(x_{dom}\cos(2\pi\theta) + y_{dom}\sin(2\pi\theta), y_{dom}\cos(2\pi\theta) - x_{dom}\sin(2\pi\theta))$. ◀

A.3 Proofs for Section 5

We now show that B_ψ^ℓ is fully in P_{dom}, tangent to the surface of P_{dom}, for $\mathbf{d} = (2,2,0,0,0,2\pi\ell)$.

▶ **Lemma 20.** *Let* $\mathbf{d} = (2,2,0,0,0,2\pi\ell)$*. For all* $\mathbf{d}' \neq \mathbf{d}$ *with* $\mathbf{d}' \in \mathcal{B}_\psi^\ell$*, we have* $\mathbf{d}'$ *is strictly in* P_{dom}*, ie for all* n $v_n^{dom}(\mathbf{d}') > 0$*.*

Proof. Let $\mathbf{d}' = (2 + \psi + z'_{dom}, 2 - \psi + x'_{dom}, y_{dom}, z_{res}, x_{res}, 2\pi\ell + y'_{res}) \in \mathcal{B}_\psi^\ell \setminus \{\mathbf{d}\}$. We now show that $\mathbf{d}'$ is strictly in P_{dom}, i.e. $(2 + \psi + z'_{dom}) - \sqrt{(2 - \psi + x'{}_{dom})^2 + y_{dom}^2} > 0$.

We have ${z'_{dom}}^2 + {x'_{dom}}^2 + y_{dom}^2 \leq 2\psi^2$, and we write ${z'_{dom}}^2 + {x'_{dom}}^2 + y_{dom}^2 = 2\lambda_1^2\psi^2$, with $\lambda_1 \in [0,1]$ and $\lambda_1 = 1$ iff $(z_{res}, x_{res}, y'_{res}) = (0,0,0)$. We also write ${x'_{dom}}^2 + y_{dom}^2 = 2\lambda_2^2\lambda_1^2\psi^2$ with $\lambda_2^2 \in [0,1]$, i.e. ${z'_{dom}}^2 = 2(1 - \lambda_2^2)\lambda_1^2\psi$. We write ${x'_{dom}}^2 = 2\lambda_3^2\lambda_2^2\lambda_1^2\psi^2$, with $\lambda_3 \in [0,1]$ and $\lambda_3 = 1$ iff $y_{dom} = 0$. We write $x'_{dom} = \sqrt{2}\lambda_3\lambda_2\lambda_1\psi$ and $z'_{dom} = \pm\sqrt{2}\sqrt{1 - \lambda_2^2}\lambda_1\psi$. That is, $\mathbf{d}$ is the configuration with $\lambda_1 = \lambda_3 = 1$ and $\lambda_2 = \frac{\sqrt{2}}{2}$.

We have $(2 - \psi + x'{}_{dom})^2 + y_{dom}^2 = (2 - \psi)^2 + x'^2_{dom} + y_{dom}^2 + 2(2 - \psi)x'{}_{dom} = (2 - \psi)^2 + 2\lambda_2^2\lambda_1^2\psi^2 + 2\sqrt{2}(2 - \psi)\lambda_3\lambda_2\lambda_1\psi \leq (2 - \psi)^2 + 2\lambda_2^2\lambda_1^2\psi^2 + 2\sqrt{2}(2 - \psi)|\lambda_2|\lambda_1\psi = (2 - \psi + \sqrt{2}|\lambda_2|\lambda_1\psi)^2$, with equality iff $\lambda_3 = 1$, ie when $y_{dom} = 0$. Given that $\psi < \frac{1}{3}$, we have $2 - \psi + \sqrt{2}|\lambda_2|\lambda_1\psi > 0$.

Thus $(2 + \psi + z'_{dom}) - \sqrt{(2 - \psi - x'{}_{dom})^2 + y_{dom}^2} \geq 2 + \psi \pm \sqrt{2}\sqrt{1 - \lambda_2^2}\lambda_1\psi - (2 - \psi + \sqrt{2}|\lambda_2|\lambda_1\psi) = 2\psi - \sqrt{2}(|\lambda_2| \pm \sqrt{1 - \lambda_2^2})\lambda_1\psi \geq 2\psi - 2\lambda_1\psi \geq 0$ as $(|\lambda_2| \pm \sqrt{1 - \lambda_2^2}) \leq \sqrt{2}$, with equality iff $\lambda_2 = \pm\frac{\sqrt{2}}{2}$. That is, for all $d' \in \mathcal{B}_\psi^\ell$, $d' \in P_{dom}$, and it is strictly inside whenever $\mathbf{d}' \neq \mathbf{d}$ (one can check that $\lambda_1 = \lambda_3 = 1$ and $\lambda_2 = -\frac{\sqrt{2}}{2}$ does not yield the overall equality). ◀

A.4 Results and Proofs for Section 6

A deep result of Masser [19] shows that integer multiplicative relationships between algebraic numbers can be elicited efficiently.

▶ **Theorem 21** (Masser [19]). *Let k, be fixed, and let $e^{i\theta_1}, ..., e^{i\theta_k}$ be complex algebraic numbers of unit modulus. Consider the free abelian group L under addition given by $L = \{(\lambda_1, ..., \lambda_k) \in \mathbb{Z}^k : e^{i\lambda_1\theta_1}...e^{i\lambda_k\theta_k} = 1\}$. L has a basis $\{\mathbf{l_1}, ..., \mathbf{l_p}\} \subset \mathbb{Z}^k$ with $p \leq k$. The basis can be computed in time polynomial and each entry in the basis vector is polynomially bounded in size$(e^{i\theta_1}), ...,$ size$(e^{i\theta_k})$.*

Kronecker theorem [11] states that each linear combination $\mathbf{t}$ of the basis given by Masser theorem can be approximated by a power $\mathbf{s}^n$ of $\mathbf{s} = (e^{i\theta_1}, \ldots, e^{i\theta_k})$.

▶ **Theorem 22** (Kronecker [11]). *Let $\theta_1, ..., \theta_k, \phi_1, ..., \phi_k \in [0, 2\pi)$. The following two statements are equivalent:*

- *For any $\epsilon' > 0$, there exist $n, m_1, ..., m_k \in \mathbb{Z}$ such that for $1 \leq j \leq k$ we have $|n\theta_j - \phi_j - 2m_j\pi| \leq \epsilon'$*
- *For every tuple $(\lambda_1, ...\lambda_k)$ of integers such that $\sum_{j=1}^{k} \lambda_j\theta_j \in 2\pi\mathbb{Z}$ we have $\sum_{j=1}^{k} \lambda_j\phi_j \in 2\pi\mathbb{Z}$*

Finally, we provide the reasoning why constant C independent of n exists which bounds the distance, i.e., we prove Lemma 19.

▶ **Lemma 19.** *There exists C such that for all n, distance$(\mathbf{c}, H_n) \leq C \cdot |v_n(\mathbf{c})|$.*

Proof. Let $n \in \mathbb{N}$. We have *distance*$(\mathbf{c}, H_n) = \frac{|u_n(\mathbf{c})|}{||\mathbf{y}||}$ for $\mathbf{y}$ the first row of $\mathbf{M}^n$ by basic geometry. Let H be the transformation matrix between the basis of initial configurations and the basis of the exponential polynomial solution of $(u_n)_{n\in\mathbb{N}}$. Let $\mathbf{x} = (x_1, \ldots, x_\kappa)$ with $x_i = n^k\rho_j^n$ so that to cover every root ρ_j and multiplicities $k = 1, \ldots, m_j$. We have $u_n(\mathbf{c}) = \mathbf{y} \cdot \mathbf{c} = \mathbf{x} \cdot (H \cdot \mathbf{c})$ for all initial configurations $\mathbf{c}$, i.e., $\mathbf{y} = \mathbf{x} \cdot H$. That is, there exists a constant $D > 0$ depending upon H with $||\mathbf{y}|| \geq Dn^m\rho^n$ for ρ the modulus of a dominant root and $m + 1$ the highest multiplicity of a root of modulus ρ. We obtain *distance*$(\mathbf{c}, H_n) \leq \frac{|u_n(\mathbf{c})|}{Dn^m\rho^n} = \frac{|v_n(\mathbf{c})|}{D}$. ◀

Finally, we provide Algorithm 1 for ∃-robust Skolem.

Algorithm 1 Robust Skolem.

```
Data: Companion matrix $\mathbf{M} \in \mathbb{Q}^{\kappa\times\kappa}$ of $(u_n)_{n\in\mathbb{N}}$ and center of ball $\mathbf{c}_0 \in \mathbb{Q}^\kappa$
$\{\gamma_j\}_j \leftarrow$ eigenvalues of $\mathbf{M}$,   $\rho \leftarrow \max_j |\gamma_j|$,   $\{e^{i\theta_j}\}_{j=1}^{k} \leftarrow \{\gamma_i/\rho \mid |\gamma_i| = \rho\}$
Determine $T$ Torus obtained by applying Masser's result (Theorem 21) to $\{\theta_j\}_{j=1}^{k}$
$\nu \leftarrow \min_{\mathbf{t}\in T} |\mathrm{dominant}(\mathbf{c}, \mathbf{t})|$ (Proposition 17)
if $\nu = 0$ then
    return NO (Proposition 18)
else
    Compute $N$ such that $v_n^{res}(\mathbf{c}_0) < \frac{\nu}{2}$ for all $n > N$
    foreach $n \in \{0, 1, \ldots, N\}$ do
        if $v_n(\mathbf{c}_0) = 0$ then
            return NO
        end
    end
    return YES
end
```

Approximability of Robust Network Design: The Directed Case

Yacine Al-Najjar[1]
Huawei Technologies, Paris Research Center, France
Samovar, Telecom SudParis, Institut Polytechnique de Paris, France

Walid Ben-Ameur
Samovar, Telecom SudParis, Institut Polytechnique de Paris, France

Jérémie Leguay
Huawei Technologies, Paris Research Center, France

Abstract

We consider robust network design problems where an uncertain traffic vector belonging to a polytope has to be dynamically routed to minimize either the network congestion or some linear reservation cost. We focus on the variant in which the underlying graph is directed. We prove that an $O(\sqrt{k}) = O(n)$-approximation can be obtained by solving the problem under static routing, where k is the number of commodities and n is the number of nodes. This improves previous results of Hajiaghayi et al. [SODA'2005] and matches the $\Omega(n)$ lower bound of Ene et al. [STOC'2016] and the $\Omega(\sqrt{k})$ lower bound of Azar et al. [STOC'2003]. Finally, we introduce a slightly more general problem version where some flow restrictions can be added. We show that it cannot be approximated within a ratio of $k^{\frac{c}{\log\log k}}$ (resp. $n^{\frac{c}{\log\log n}}$) for some constant c. Making use of a weaker complexity assumption, we prove that there is no approximation within a factor of $2^{\log^{1-\epsilon} k}$ (resp. $2^{\log^{1-\epsilon} n}$) for any $\epsilon > 0$.

2012 ACM Subject Classification Theory of computation; Theory of computation → Data structures design and analysis; Theory of computation → Approximation algorithms analysis; Theory of computation → Routing and network design problems

Keywords and phrases Robust Optimization, Network Design, Approximation, Inapproximability, Competitive Ratio of Oblivious Routing

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.6

1 Introduction

Network optimization [38, 28] plays a crucial role for telecommunication operators since it permits to carefully invest in infrastructures. As the traffic is continuously increasing, the network's capacity needs to be expanded through careful investments every year. However, the dynamic nature of the traffic due to ordinary daily fluctuations, long term evolution and unpredictable events requires to consider uncertainty on the traffic demand when dimensioning network resources. In this context, we provide new approximability results on two tightly related variants of the robust network design problem, the minimization of either the congestion or a linear cost.

Let us consider a directed graph $G = (V(G), E(G))$ representing a communication network. The traffic is characterized by a set of commodities $h \in \mathcal{H}$ associated to different node pairs and traffic values d_h. The demand vector $d = (d_h)_{h\in\mathcal{H}}$ is assumed to be uncertain and more precisely to belong to a polyhedral set $\mathcal{D}$. The *polyhedral model* was introduced in [6, 7] as an extension of the *hose model* [14, 17], where limits on the total traffic going into (resp. out of) a node are considered.

[1] corresponding author

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 6; pp. 6:1–6:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The routing of a commodity h can be represented by a unit flow (also called routing template) $f_{h,.} = (f_{h,e})_{e \in E(G)}$ from the source $s(h)$ to the sink $t(h)$.

When solving a robust network design problem, several objective functions can be considered. Given a capacity c_e for each edge e, one might be interested in minimizing the congestion given by $\max_{e \in E(G)} \frac{u_e}{c_e}$ where u_e is the reserved capacity on edge e. Another common objective function is given by the linear reservation cost $\sum_{e \in E(G)} \lambda_e u_e$. This can also represent the average congestion by taking $\lambda_e = \frac{1}{c_e}$. The goal is to choose a reservation vector u so that the network is able to support any demand vector $d \in \mathcal{D}$, i.e., there exists a (fractional) routing serving every commodity such that the total flow on each edge e is less than the reservation u_e.

The robust network design problem that we are focusing on in this paper, is referred to as *dynamic routing* in the literature since the network is optimized such that any realization of traffic vector in the uncertainty set has its own routing (i.e., $f_{h,.}$ depends on d). The robust network design problem where a linear reservation cost is minimized was proved to be co-NP hard in [21] when the graph is directed. A stronger co-NP hardness result is given in [12] where the graph is undirected (this implies the directed case result). Some exact solution methods for robust network design have been considered in [13, 30]. Some special cases where dynamic routing is easy to compute have been described in [8, 18, 31]. For each of the two problems of congestion minimization and linear reservation cost minimization under dynamic routing, it is proved in [1] that the optimal value cannot be approximated within any constant (unless $P = NP$) and within $\Omega(\frac{\log n}{\log \log n})$ (under ETH assumption) for an undirected graph having n vertices. This leads again to the same inapproximability result for the directed case.

Routing with uncertain demands has received a significant interest from the community. As opposed to dynamic routing, *static routing* or *stable routing* was introduced in [6]: it consists in choosing a fixed flow $f_{h,.}$ of value 1 for each commodity h. Static routing is also called *oblivious routing* in [2, 3]. In this case, polynomial-time algorithms to compute optimal static routing (with respect to either congestion or linear reservation cost) have been proposed [2, 3, 6, 7] based on either duality or cutting-plane algorithms.

To further improve solutions of static routing and overcome complexity issues related to dynamic routing, a number of restrictions on routing have been considered to design polynomial-time algorithms. This includes, for example, the approaches proposed in [5, 9, 27, 34, 35, 39].

Most of the literature studied the undirected case of the robust network design problem while only a few papers, such as [3, 7, 21, 24], address the directed case. In this work, we mainly focus on the approximability of robust network design problems under dynamic routing in directed networks, while minimizing either congestion or some linear reservation cost.

In the rest of this section, we summarize the main contributions of the paper and their positioning with regard to prior work. Then, we review some related state of the art.

1.1 Our results

- We prove that compared to dynamic routing, when static routing is considered, congestion is multiplied by a factor less than or equal to $\sqrt{8k}$ where k is the number of commodities. This implies that the gap between static routing and dynamic routing for the congestion minimization problem is $O(\sqrt{k}) = O(n)$ where n is the number of nodes. The best-known previous bound is $O(\sqrt{k} n^{\frac{1}{4}} \log n)$ and was given by [24]. The same $\sqrt{8k}$ bound applies to the linear reservation cost problem. The new upper bound matches the $\Omega(\sqrt{k})$ lower bound of [3] and the $\Omega(n)$ lower bound of [16].

- We introduce a more general version of the two robust network design problems (related to congestion and linear cost) by considering some flow restrictions (each commodity h can only be routed through edges inside a given subset E_h). The upper bound $\sqrt{8k} = O(n)$ is still valid and the static versions of the problems can still be solved in polynomial-time. We show some strong inapproximability results for this problem. More precisely, we prove that unless $NP \subseteq SUBEXP$, neither minimum dynamic congestion nor optimal linear cost can be approximated within a ratio of $k^{\frac{c}{\log\log k}}$ (resp. $n^{\frac{c}{\log\log n}}$) for some constant c. Making use of a weaker assumption, we get that unless $NP \subseteq QP$, there is no approximation within a factor of $2^{\log^{1-\epsilon} k}$ (resp. $2^{\log^{1-\epsilon} n}$) for any $\epsilon > 0$. This result improves the $\Omega(\frac{\log n}{\log\log n})$ inapproximability bound of [1] for the undirected case that also applies to the directed one.

1.2 Related work

Let us first consider that the graph is undirected and a linear cost is minimized. A result attributed to A. Gupta ([11], see also [19] for a more detailed presentation) leads to an $O(\log n)$ approximation algorithm for linear cost under dynamic fractional routing. Furthermore, this approximation is achieved by a routing on a (fixed) single tree. In particular, this shows that the ratios between the dynamic and the static solutions under fractional routing ($\frac{\text{lin}_{\text{sta-frac}}}{\text{lin}_{\text{dyn-frac}}}$) (lin denotes here the optimal linear cost of the solution) and between single path and fractional routing under the static model ($\frac{\text{lin}_{\text{sta-sing}}}{\text{lin}_{\text{sta-frac}}}$) is in $O(\log n)$ and provides an $O(\log n)$ approximation for static single path routing $\text{lin}_{\text{sta-sing}}$. On the other hand [33] shows that the static single path problem cannot be approximated within a $\Omega(\log^{\frac{1}{4}-\epsilon} n)$ ratio unless $NP \not\subseteq ZPTIME(n^{polylog(n)})$. As noticed in [19], this implies (assuming this complexity conjecture) that the gap $\frac{\text{lin}_{\text{sta-sing}}}{\text{lin}_{\text{sta-frac}}}$ is in $\Omega(\log^{\frac{1}{4}-\epsilon} n)$. [19] has shown that the gap $\frac{\text{lin}_{\text{sta-frac}}}{\text{lin}_{\text{dyn-frac}}}$ is $\Omega(\log n)$.

For the linear cost objective function and undirected graphs, an extensively studied polyhedron is the *symmetric* hose model. The demand vector is here not oriented (i.e, there is no distinction between a demand from i to j and a demand from j to i), and uncertainty is defined by considering an upper-bound limit b_i for the sum of demands related to node i. A 2-approximation has been found for the dynamic fractional case [17, 21] based on tree routing (where we route through a static tree that should be found) showing that $\frac{\text{lin}_{\text{sta-tree}}}{\text{lin}_{\text{dyn-frac}}} \leq 2$. It has been conjectured that this solution resulted in an optimal solution for the static single path routing. This question has been open for some time and has become known as the *VPN conjecture*. It was finally answered by the affirmative in [20]. The *asymmetric* hose polytope was also considered in many papers. An approximation algorithm is proposed to compute $\text{lin}_{\text{sta-sing}}$ within a ratio of 3.39 [15] (or more precisely 2 plus the best approximation ratio for the Steiner tree problem). If $\mathcal{D}$ is a *balanced* asymmetric hose polytope, i.e., $\sum_{v \in V} b_v^{out} = \sum_{v \in V} b_v^{in}$ where b_v^{in} (resp. b_v^{out}) is the upper bound for the traffic entering into (resp. going out of) v, then the best approximation factor becomes 2 [15]. Moreover, if we assume that $b_v^{out} = b_b^{in}$, then $\text{lin}_{\text{sta-sing}}$ is easy to compute and we get that $\text{lin}_{\text{sta-tree}} = \text{lin}_{\text{sta-sing}}$ [32]. In other words, there is some similarity with the case where $\mathcal{D}$ is a symmetric hose polytope.

When congestion is considered, [36] proved the existence of an oblivious (or static) routing with a competitive ratio of $O(\log^3 n)$ with respect to optimum routing of any traffic matrix. Then, [25] improved the bound to $O(\log^2 n \log\log n)$ and gave a polynomial-time algorithm to find such a static routing. Finally, [37] described an $O(\log n)$ approximation algorithm for static routing with minimum congestion. Notice that the bound given by static routing cannot provide a better bound than $O(\log n)$ since a lower bound of $\Omega(\log n)$ is achieved by static routing for planar graphs [29, 4]. It has also been shown in [23] that the gap between the dynamic fractional routing and a dynamic fractional routing restricted to a polynomial number of paths can be $\Omega(\frac{\log n}{\log\log n})$.

In a recent study on the approximability of robust network design [1] for the undirected case, it was proved that minimum dynamic congestion and the optimal linear cost cannot be approximated within any constant factor. Then using the ETH conjecture, it is shown there that they cannot be approximated within $\Omega(\frac{\log n}{\log\log n})$. This implies that the well-known $O(\log n)$ approximation ratio established in [37] is tight. Using a Lagrange relaxation approach, it is also shown in [1] that any α-approximation algorithm for the robust network design problem with linear reservation costs directly leads to an α-approximation for the problem of minimum congestion. This is used there to prove in a different way the $O(\log n)$ result of [37] starting from the one of [22] (attributed to A. Gupta and related to the linear cost minimization).

The closest papers to ours are [3, 16, 24]. When a directed graph is considered and congestion is minimized, [3] has shown that the gap between static fractional routing and dynamic fractional routing can be $\Omega(\sqrt{k})$ while [24] proves that the gap is upper-bounded by $O(\sqrt{k}n^{\frac{1}{4}}\log n)$. Since the instance provided in [3] contains vertices with large degree, [24] studied the version where the degree is less than some constant and all commodities have the same sink. An instance with a $\Omega(\sqrt{n})$ gap was then provided in [24], while the upper bound becomes $O(\sqrt{n}\log n)$. [24] considered also the case of symmetric demands (in that paper, symmetry means that for any two nodes u and v, the demand from u to v is equal to the demand from v to u) and shows that the upper bound of the static to dynamic ratio becomes $O(\sqrt{k}\log^{5/2} n)$. A general $\Omega(n)$ lower bound was later proposed in [16]. They also introduced the notion of balance for directed graphs. A weighted directed graph is α-balanced if for every subset $S \subseteq V$, the total weight of edges going from S to $V\backslash S$ is within a factor α of the total weight of edges directed from $V\backslash S$ to S. Using this new parameter, they show that for single source instances an upper bound of $O(\alpha\frac{\log^3 n}{\log\log n})$ holds for the competitive ratio of static routing.

2 Preliminaries

In this section, we give more formal definitions of the *robust network design* problems considered in this paper. Some notation and basic results are also recalled. The *congestion minimization* variant takes as input a graph $G = (V, E)$, a vector of link capacities $c \in \mathbb{R}^E_+$ and a set of commodities $\mathcal{H}$. Each commodity $h \in \mathcal{H}$ has a source $s(h)$ and destination $t(h)$ in V. We also have as input a polytope $\mathcal{D}$ of all possible demand vectors $d \in \mathbb{R}^{\mathcal{H}}_+$ specifying the demand d_h that needs to be sent from $s(h)$ to $t(h)$. An instance $\mathcal{I}$ of the congestion minimization problem might be denoted by $\mathcal{I} = (G, c, \mathcal{H}, \mathcal{D})$. We use n to denote the number of nodes ($n = |V|$), while k denotes the number of commodities ($k = |\mathcal{H}|$). Given two nodes $s, t \in V$, a routing template (also called a unit flow) from s to t is a vector $f \in \mathbb{R}^E_+$ satisfying the standard flow conservation constraints. For each vertex v, $\sum\limits_{e\in\delta^+(v)} f_e - \sum\limits_{e\in\delta^-(v)} f_e$ is required to be equal to $1, -1$ or 0 when v is respectively the source s, the destination t or any other node, where $\delta^+(v)$ (resp. $\delta^-(v)$) denotes the set of edges going out of (resp. entering into) v.

A vector $f \in \mathbb{R}^{\mathcal{H}\times E}_+$ is a routing if for each commodity $h \in \mathcal{H}$, $f_{h,.} = (f_{h,e})_{e\in E}$ is a routing template from $s(h)$ to $t(h)$. The set of all possible routing schemes is denoted by $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{H}\times E}_+$. The total flow on each link $e \in E$ is $\sum\limits_{h\in\mathcal{H}} f_{h,e}d_h$ and its congestion is the total flow on the link e divided by its capacity c_e. Let $\text{cong}(f,d)$ denote the maximum congestion over all links $e \in E$, i.e. $\text{cong}(f,d) = \max\limits_{e\in E} \sum\limits_{h\in\mathcal{H}} \frac{f_{h,e}d_h}{c_e}$. Two problems can be considered depending on whether the routing can be adapted to each demand vector d in $\mathcal{D}$ or if only one fixed routing $f \in \mathcal{F}$ can be used. In the first case, the routing is said to be *dynamic*. The dynamic congestion is formally defined as: $\text{cong}_{\text{dyn}}(\mathcal{I}) = \max\limits_{d\in\mathcal{D}} \min\limits_{f\in\mathcal{F}} \text{cong}(f,d)$. In the second case, the routing is said to be *static* (or oblivious). This static congestion is formally defined as: $\text{cong}_{\text{sta}}(\mathcal{I}) = \min\limits_{f\in\mathcal{F}} \max\limits_{d\in\mathcal{D}} \text{cong}(f,d)$. Notice that when clear from the context, we might use notation $\text{cong}_{\text{dyn}}(\mathcal{D})$ and $\text{cong}_{\text{sta}}(\mathcal{D})$ to insist on the dependency on $\mathcal{D}$ when all other parameters of the instance $\mathcal{I}$ are fixed.

In the same way, we can define the robust linear reservation problem. As already said in Section 1, given a positive cost vector $(\lambda_e)_{e\in E}$, we aim to reserve a capacity $u_e \geq 0$ on each link e such that $\sum_{e\in E} \lambda_e u_e$ is minimized and $\sum\limits_{h\in\mathcal{H}} f_{h,e}d_h \leq u_e$ holds for any demand vector d. An instance can then be denoted by $(G, \lambda, \mathcal{H}, \mathcal{D})$. We also have two variants depending on routing. The optimal cost is then denoted by $\text{lin}_{\text{dyn}}(\mathcal{I})$ (or $\text{lin}_{\text{dyn}}(\mathcal{D})$) and $\text{lin}_{\text{sta}}(\mathcal{I})$ (or $\text{lin}_{\text{sta}}(\mathcal{D})$). Notice that only fractional routing is considered in this paper (this is why the subscript *frac* used in Section 1.2 is omitted in the rest of the paper).

For concise notation, the four variants of the robust optimization problems considered in this paper will simply be denoted by lin_{sta}, lin_{dyn}, cong_{sta} and cong_{dyn}.

All previous definitions still make sense even when $\mathcal{D}$ is not a polytope. However, the next lemma tells us that the optimal objective value does not increase when the uncertainty set $\mathcal{S}$ is replaced by its convex-hull (this lemma can be considered as a folklore result that is implicitly used in many robust optimization papers).

▶ **Lemma 1.** *Let $\mathcal{S} \subset \mathbf{R}_+^{\mathcal{H}}$ be a compact set. Then $cong_{sta}(\mathcal{S}) = cong_{sta}(conv(\mathcal{S}))$, $cong_{dyn}(\mathcal{S}) = cong_{dyn}(conv(\mathcal{S}))$, $lin_{sta}(\mathcal{S}) = lin_{sta}(conv(\mathcal{S}))$, and $lin_{dyn}(\mathcal{S}) = lin_{dyn}(conv(\mathcal{S}))$.*

Proof. Since $\mathcal{S} \subseteq \text{conv}(\mathcal{S})$, we have $\text{cong}_{\text{sta}}(\mathcal{S}) \leq \text{cong}_{\text{sta}}(\text{conv}(\mathcal{S}))$ and $\text{cong}_{\text{dyn}}(\mathcal{S}) \leq \text{cong}_{\text{dyn}}(\text{conv}(\mathcal{S}))$. The same holds for the robust linear cost problem. Moreover, given a static routing solution f and the corresponding reservation vector u, we have $\sum\limits_{h\in\mathcal{H}} f_{h,e}d_h \leq u_e$ for any $d \in \mathcal{S}$. Consider any point d' of $\text{conv}(\mathcal{S})$ written as $d' = \sum_{d\in\mathcal{S}} \alpha_d d$ ($\alpha_d \geq 0, \sum_{d\in\mathcal{S}} \alpha_d = 1$). By multiplying the previous inequalities by α_d and summing them all, we get that $\sum\limits_{h\in\mathcal{H}} f_{h,e}d'_h \leq u_e$ implying that f and u are feasible. Therefore, we have $\text{lin}_{\text{sta}}(\mathcal{S}) = \text{lin}_{\text{sta}}(\text{conv}(\mathcal{S}))$. The proof can be easily extended to the dynamic routing version and to the congestion objective function. ◀

Let us now focus on the connection between the congestion problem and the linear cost problem. The first proposition is from [19] and states that if the static to dynamic ratio is less than or equal to α for the congestion problem, then the same applies to the robust linear reservation problem.

▶ **Proposition 2** ([19]). *Let $\mathcal{I} = (G, c, \mathcal{H}, \mathcal{D})$ and assume that $cong_{sta}(\mathcal{I}) \leq \alpha\ cong_{dyn}(\mathcal{I})$ for some $\alpha \geq 1$ and for* ***any*** *vector $c \in \mathbb{R}_+^{\mathcal{H}}$. Then $lin_{sta}(\mathcal{I}') \leq \alpha\ lin_{dyn}(\mathcal{I}')$ where $\mathcal{I}' = (G, \lambda, \mathcal{H}, \mathcal{D})$ for* ***any*** *cost vector $\lambda \in \mathbb{R}_+^{\mathcal{H}}$.*

Proof. Given a cost vector λ, let $c^*_{\text{dyn}} \in \mathbb{R}_+^E$ be the reservation vector (i.e., u) obtained when the linear cost is minimized under dynamic routing. Let then $\mathcal{I} = (G, c^*_{\text{dyn}}, \mathcal{H}, \mathcal{D})$. We clearly have $\text{cong}_{\text{dyn}}(\mathcal{I}) \leq 1$ and $\text{cong}_{\text{sta}}(\mathcal{I}) \leq \alpha\ \text{cong}_{\text{dyn}}(\mathcal{I}) \leq \alpha$. Therefore, $\alpha\ c^*_{\text{dyn}}$ is a feasible reservation vector for the lin_{sta} problem related to instance $\mathcal{I}' = (G, \lambda, \mathcal{H}, \mathcal{D})$ and its cost is α times the cost of c^*_{dyn}. ◀

A converse result is presented in [1]. While the proof in [1] was given in the context of undirected graphs, it can be repeated verbatim for the directed case (the proof is based on a Lagrange relaxation approach and a careful application of the ellipsoid method).

▶ **Proposition 3** ([1]). *Let $\mathcal{I}' = (G, \lambda, \mathcal{H}, \mathcal{D})$ and assume that $lin_{sta}(\mathcal{I}') \leq \alpha\ lin_{dyn}(\mathcal{I}')$ for some $\alpha \geq 1$ and for **any** cost vector $\lambda \in \mathbb{R}^{\mathcal{H}}_{+}$. Then $cong_{sta}(\mathcal{I}) \leq \alpha\ cong_{dyn}(\mathcal{I})$ where $\mathcal{I} = (G, c, \mathcal{H}, \mathcal{D})$ for **any** capacity vector $c \in \mathbb{R}^{\mathcal{H}}_{+}$. Moreover, any β-approximation ($\beta \geq 1$) for lin_{dyn} leads to a β-approximation for $cong_{dyn}$.*

To close this section, let us recall some notation and assumptions that will be used in the rest of the paper. The uncertainty set (i.e., the set of demand vectors) $\mathcal{D}$ is assumed to be polyhedral and down monotone (i.e., if $d \in \mathcal{D}$, then $d' \in \mathcal{D}$ for any $0 \leq d' \leq d$). Let $d^{max}(\mathcal{D})$ be the vector representing the maximum commodity values (i.e., $d^{max}_h(\mathcal{D}) = \max_{d\in\mathcal{D}} d_h$). We will naturally assume that $d^{max}_h > 0$ for any $h \in \mathcal{H}$ since otherwise the commodity can just be ignored. When the polytope $\mathcal{D}$ is clear from the context, we just write d^{max} (instead of $d^{max}(\mathcal{D})$).

Let I, J be some set of indices. For a vector $v \in \mathbb{R}^{I\times J}$ and $i \in I$ we denote by $v_{i,.}$ the vector $w \in \mathbb{R}^J$ defined by $w_j = v_{i,j}$. Given a set $X \in \mathbb{R}^I$ and $\lambda \geq 0$, we denote by λX the set $\{\lambda x | x \in X\}$.

3 Approximation of dynamic congestion by static congestion

We are going to prove Theorem 4 stating that compared to dynamic routing, when static routing is considered, congestion is multiplied by a factor less than or equal to $\sqrt{8k}$. This result improves the upper bound $O(\sqrt{k}n^{1/4}\log n)$ from [24]. It implies that the gap between static and dynamic congestion is $O(\sqrt{k}) = O(n)$. By combining Proposition 2 with Theorem 4, we also obtain similar results for the minimization of a linear reservation cost, i.e., that $\mathrm{lin}_{\mathrm{sta}}(\mathcal{D}) \leq \sqrt{8k}.\mathrm{lin}_{\mathrm{dyn}}(\mathcal{D})$.

▶ **Theorem 4.** *$cong_{sta}(\mathcal{D}) \leq \sqrt{8k}.cong_{dyn}(\mathcal{D})$. Therefore $\frac{cong_{sta}(\mathcal{D})}{cong_{dyn}(\mathcal{D})} = O(n)$.*

To derive an upper bound for the ratio $\mathrm{cong}_{\mathrm{sta}}(\mathcal{D})/\mathrm{cong}_{\mathrm{dyn}}(\mathcal{D})$, our strategy first consists in approximating the uncertainty set either by a box or a simplex where $\mathrm{cong}_{\mathrm{sta}}(\mathcal{D}) = \mathrm{cong}_{\mathrm{dyn}}(\mathcal{D})$. While this method yields an $O(k)$ upper bound, we obtain further improvement by partitioning the set of commodities into two sets $\mathcal{H}_1$, $\mathcal{H}_2$ and considering a box approximation for $\mathcal{D}_1$ and a simplex approximation for $\mathcal{D}_2$, where $\mathcal{D}_1$ and $\mathcal{D}_2$ are respectively the projections of $\mathcal{D}$ on $\mathbb{R}^{\mathcal{H}_1}$ and $\mathbb{R}^{\mathcal{H}_2}$.

To prove Theorem 4, we first present some preliminary lemmas.

Lemma 5 states that if the uncertainty set $\mathcal{D}$ can be well approximated by another set $\mathcal{D}'$ for which $\mathrm{cong}_{\mathrm{sta}}(\mathcal{D}') = \mathrm{cong}_{\mathrm{dyn}}(\mathcal{D}')$, then $\mathrm{cong}_{\mathrm{sta}}(\mathcal{D})$ gives a good approximation of $\mathrm{cong}_{\mathrm{dyn}}(\mathcal{D})$.

▶ **Lemma 5.** *Let $\mathcal{D}$ and $\mathcal{D}'$ be two compact subsets of $\mathbb{R}^{\mathcal{H}}_{+}$ and $\alpha \in \mathbb{R}_+$ such that $\mathcal{D}' \subseteq \mathcal{D} \subseteq \alpha\mathcal{D}'$ and $cong_{sta}(\mathcal{D}') = cong_{dyn}(\mathcal{D}')$. Then $cong_{sta}(\mathcal{D}) \leq \alpha \cdot cong_{dyn}(\mathcal{D})$.*

Proof. The proof of this lemma relies on two simple facts. The first one is that if we scale the demand values by a factor α, then the congestion (either static or dynamic) is also scaled by the same factor α. The second fact is that $\mathrm{cong}_{\mathrm{dyn}}$ and $\mathrm{cong}_{\mathrm{sta}}$ are increasing in $\mathcal{D}$. In other words, if $\mathcal{D}_1$ and $\mathcal{D}_2$ are two subsets of $\mathbb{R}^{\mathcal{H}}_{+}$ such that $\mathcal{D}_1 \subseteq \mathcal{D}_2$, then

$\text{cong}_{\text{dyn}}(\mathcal{D}_1) \leq \text{cong}_{\text{dyn}}(\mathcal{D}_2)$ and $\text{cong}_{\text{sta}}(\mathcal{D}_1) \leq \text{cong}_{\text{sta}}(\mathcal{D}_2)$. Combining the two facts, we can write the following:

$$\text{cong}_{\text{sta}}(\mathcal{D}) \leq \text{cong}_{\text{sta}}(\alpha\mathcal{D}') = \alpha \cdot \text{cong}_{\text{sta}}(\mathcal{D}') = \alpha \cdot \text{cong}_{\text{dyn}}(\mathcal{D}') \leq \alpha \cdot \text{cong}_{\text{dyn}}(\mathcal{D}) \tag{1}$$

◀

We now provide in Lemmas 6 and 7 two classes of polytopes, based on box and simplex sets, for which $\text{cong}_{\text{sta}}(\mathcal{D}) = \text{cong}_{\text{dyn}}(\mathcal{D})$.

For a vector $d^{max} \in \mathbb{R}^{\mathcal{H}}_{+}$, let $\mathcal{B}(d^{max})$ be the box set defined by $\{d \in \mathbb{R}^{\mathcal{H}} |\ 0 \leq d \leq d^{max}\}$.

▶ **Lemma 6.** *Let $\mathcal{D} = \mathcal{B}(d^{max})$ for some $d^{max} \in \mathbb{R}^{\mathcal{H}}_{+}$. Then $cong_{dyn}(\mathcal{D}) = cong_{sta}(\mathcal{D})$.*

Proof. For a routing $f \in \mathcal{F}$ and a demand vector $d \in \mathcal{D}$, we have $\text{cong}(f, d) \leq \text{cong}(f, d^{max})$. Since $d^{max} \in \mathcal{D}$, it implies that $\max\limits_{d \in \mathcal{D}} \text{cong}(f, d) = \text{cong}(f, d^{max})$. Minimizing both sides of the equality over $f \in \mathcal{F}$, we get that $\text{cong}_{\text{sta}}(\mathcal{D}) = \min\limits_{f \in \mathcal{F}} \text{cong}(f, d^{max})$. We can also write that $\min\limits_{f \in \mathcal{F}} \text{cong}(f, d) \leq \min\limits_{f \in \mathcal{F}} \text{cong}(f, d^{max})$. Taking the maximum over all $d \in \mathcal{D}$ leads to $\text{cong}_{\text{dyn}}(\mathcal{D}) = \max\limits_{d \in \mathcal{D}} \min\limits_{f \in \mathcal{F}} \text{cong}(f, d) \leq \min\limits_{f \in \mathcal{F}} \text{cong}(f, d^{max})$. Since $d^{max} \in \mathcal{D}$, the previous inequality becomes $\text{cong}_{\text{dyn}}(\mathcal{D}) = \min\limits_{f \in \mathcal{F}} \text{cong}(f, d^{max})$. ◀

For a vector $d \in \mathbb{R}^{\mathcal{H}}_{+}$, let $\Delta(d)$ be the simplex set whose vertices are the zero vector and the k vectors $d_h e_h$ where e_h denotes the vector in $\mathbb{R}^{\mathcal{H}}_{+}$ with a component of 1 for commodity h and 0 otherwise. Formally, we have $\Delta(d) = \text{conv}\left(\{d_h e_h | h \in \mathcal{D}\} \cup \{0\}\right)$.

▶ **Lemma 7.** *Let $\mathcal{D} = \Delta(d^{max})$ where $d^{max} \in \mathbb{R}^{\mathcal{H}}_{+}$. Then $cong_{dyn}(\mathcal{D}) = cong_{sta}(\mathcal{D})$.*

Proof. Assume that $\text{cong}_{\text{dyn}}(\mathcal{D})$ has been computed and consider the obtained dynamic routing. The extreme points of $\mathcal{D}$ are the demand vectors $\{d^{max}_h e_h | h \in \mathcal{H}\} \cup \{0\}$. For each demand vector $d^{max}_h e_h$, we consider the flow $f_{h,.}$ representing its routing. Let us build a static routing f just by routing each commodity h in accordance to $f_{h,.}$. By construction, taking the extreme points of $\mathcal{D}$, we have $\text{cong}_{\text{sta}}(\{d^{max}_h e_h | h \in \mathcal{H}\} \cup \{0\}) = \text{cong}_{\text{dyn}}(\{d^{max}_h e_h | h \in \mathcal{H}\} \cup \{0\})$. By considering the convex-hulls and applying Lemma 1, we get that $\text{cong}_{\text{dyn}}(\mathcal{D}) = \text{cong}_{\text{sta}}(\mathcal{D})$. ◀

Let $\alpha_1(\mathcal{D}) = \max\limits_{d \in \mathcal{D}} \sum\limits_{h \in \mathcal{H}} \frac{d_h}{d^{max}_h}$ (remember that $d^{max}_h = \max_{d \in \mathcal{D}} d_h$). It is then clear that $\Delta(d^{max}) \subseteq \mathcal{D} \subseteq \alpha_1(\mathcal{D})\Delta(d^{max})$. Consider the box $\mathcal{B}(d^{max})$ and let $\alpha_2(\mathcal{D})$ be the smallest factor α such that d^{max}/α belongs to $\mathcal{D}$. In other words, $\alpha_2(\mathcal{D})$ represents the best approximation ratio that can be obtained through boxes. We obviously have $\frac{1}{\alpha_2(\mathcal{D})}\mathcal{B}(d^{max}) \subseteq \mathcal{D} \subseteq \mathcal{B}(d^{max})$. Figure 1 illustrates the approximations by boxes and simplices for a 2-dimensional demand polytope $\mathcal{D}$.

Since $\frac{1}{k}\mathcal{B}(d^{max}) \subseteq \Delta(d^{max}) \subseteq \mathcal{D} \subseteq \mathcal{B}(d^{max})$, $\alpha_2(\mathcal{D})$ is always less than or equal to k. And by definition, $\alpha_1(\mathcal{D})$ is also less than or equal to k.

It is easy to check that the upper bound k is reached since $\alpha_1(\mathcal{B}(d^{max})) = k$ and $\alpha_2(\Delta(d^{max})) = k$. In other words, using box and simplex approximations with the approach above, we cannot expect to prove a better upper bound for the ratio $\text{cong}_{\text{sta}}(\mathcal{D})/\text{cong}_{\text{dyn}}(\mathcal{D})$ for arbitrary uncertainty sets.

A more refined strategy is to take the best of the two bounds $\alpha_1(\mathcal{D})$, $\alpha_2(\mathcal{D})$. The next proposition states that a better bound is obtained if $\mathcal{D}$ is permutation-invariant (i.e., by permuting the components of any vector d of $\mathcal{D}$ we always get a vector inside $\mathcal{D}$). The proof is provided in Appendix.

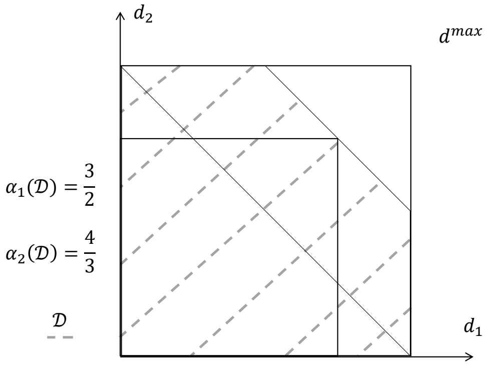

d_2
d^{max}
$\alpha_1(\mathcal{D}) = \frac{3}{2}$
$\alpha_2(\mathcal{D}) = \frac{4}{3}$
$\mathcal{D}$
d_1

Figure 1 Approximation using boxes and simplices: example of a 2-dimensional demand polytope $\mathcal{D}$.

▶ **Proposition 8.** *If $\mathcal{D}$ is permutation-invariant then* $\min\{\alpha_1(\mathcal{D}), \alpha_2(\mathcal{D})\} \leq \sqrt{k}$.

One can wonder whether a general $O(\sqrt{k})$ bound can be obtained by trying to find a better upper bound for $\min\{\alpha_1(\mathcal{D}), \alpha_2(\mathcal{D})\}$. The following example, on a specific polytope $\mathcal{D}$, shows that this is not possible. Let $\mathcal{D}$ be the product of a box $\mathcal{B}(d^1)$ of dimension $k/2$ and a simplex $\Delta(d^2)$ of the same dimension. Using the remark above we know that $\alpha_1(\mathcal{B}(d^1)) = k/2$ and $\alpha_2(\Delta(d^2)) = k/2$ implying that $\alpha_1(\mathcal{D}) \geq k/2$ and $\alpha_2(\mathcal{D}) \geq k/2$.

To overcome this difficulty, we are going to partition the set of commodities $\mathcal{H}$ into two well-chosen subsets $\mathcal{H}_1$ and $\mathcal{H}_2$, then we approximate $\mathcal{D}_1$ (resp. $\mathcal{D}_2$) defined as the projection of $\mathcal{D}$ on $\mathbb{R}^{\mathcal{H}_1}$ (resp. $\mathbb{R}^{\mathcal{H}_2}$) using a simplex (resp. a box). The algorithm used to partition the set of commodities is an adaptation of an algorithm of [10] proposed in a different context. We will also slightly improve the analysis of this algorithm ($\sqrt{8k}$ instead of $3\sqrt{k}$).

Let us start with Lemma 9 where we show how an approximation of cong_{dyn} in $\mathcal{D}$ can be obtained from cong_{sta} using the approximations related to $\mathcal{D}_1$ and $\mathcal{D}_2$.

▶ **Lemma 9.** *Let $\mathcal{H}_1, \mathcal{H}_2$ be a partition of $\mathcal{H}$ and $\mathcal{D}_1, \mathcal{D}_2$ be the projection of $\mathcal{D}$ on $\mathbb{R}^{\mathcal{H}_1}$ and $\mathbb{R}^{\mathcal{H}_2}$. Suppose that for some $\alpha_1, \alpha_2 \geq 1$ we have $cong_{sta}(\mathcal{D}_1) \leq \alpha_1 cong_{dyn}(\mathcal{D}_1)$ and $cong_{sta}(\mathcal{D}_2) \leq \alpha_2 cong_{dyn}(\mathcal{D}_2)$, then $cong_{sta}(\mathcal{D}) \leq (\alpha_1 + \alpha_2) cong_{dyn}(\mathcal{D})$.*

Proof. We first show that we have $\text{cong}_{\text{sta}}(\mathcal{D}) \leq \text{cong}_{\text{sta}}(\mathcal{D}_1) + \text{cong}_{\text{sta}}(\mathcal{D}_2)$.

$$\begin{aligned}
\text{cong}_{\text{sta}}(\mathcal{D}) &= \max_{d \in \mathcal{D}} \min_{f \in \mathcal{F}} \text{cong}(f, d) \\
&\leq \max_{d^1 \in \mathcal{D}_1, d^2 \in \mathcal{D}_2} \min_{f \in \mathcal{F}} \text{cong}(f, d^1) + \text{cong}(f, d^2) \\
&= \max_{d^1 \in \mathcal{D}_1} \min_{f \in \mathcal{F}} \text{cong}(f, d^1) + \max_{d^2 \in \mathcal{D}_2} \min_{f \in \mathcal{F}} \text{cong}(f, d^2) \\
&= \text{cong}_{\text{sta}}(\mathcal{D}_1) + \text{cong}_{\text{sta}}(\mathcal{D}_2)
\end{aligned}$$

We now prove the lemma: $\text{cong}_{\text{sta}}(\mathcal{D}) \leq \text{cong}_{\text{sta}}(\mathcal{D}_1) + \text{cong}_{\text{sta}}(\mathcal{D}_2) \leq \alpha_1 \text{cong}_{\text{dyn}}(\mathcal{D}_1) + \alpha_2 \text{cong}_{\text{dyn}}(\mathcal{D}_2) \leq (\alpha_1 + \alpha_2) \text{cong}_{\text{dyn}}(\mathcal{D})$. ◀

Let us now present Algorithm 1 that can be seen as a direct adaptation of the partitioning algorithm of [10] (Algorithm $\mathcal{A}$, Fig. 1) for our dynamic routing problem. It has initially been introduced for the analysis of affine policies in a class of two-stage adaptive linear optimization problems. The main idea of Algorithm 1 is to partition the set of commodities into two sets $\mathcal{H}_1$ and $\mathcal{H}_2$ and to produce a vector $\beta \in \mathbb{R}_+^{\mathcal{H}}$ such that $\max_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}_1} \frac{d_h}{d_h^{max}} \leq \gamma \sqrt{k}$

(i.e., $\alpha_1(\mathcal{D}_1) \leq \gamma\sqrt{k}$ for $\gamma > 0$) and $\beta_h \geq d_h^{max}$ for any $h \in \mathcal{H}_2$. The returned vector β is built as a sum of at most Z points of $\mathcal{D}$ where Z is the number of iterations of the algorithm. Since the vector $\frac{1}{Z}\beta$ belongs to $\mathcal{D}$, we deduce that $\alpha_2(\mathcal{D}_2) \leq Z$. We will show in Lemma 10 that Z is less than or equal to $2\frac{\sqrt{k}}{\gamma}$ leading to $\alpha_2(\mathcal{D}_2) \leq 2\frac{\sqrt{k}}{\gamma}$. Notice that γ is equal to 1 in the original algorithm of [10]. Let us describe more precisely the different steps of Algorithm 1. At iteration i, $\mathcal{H}_1^i, \mathcal{H}_2^i$ denote the current partitions of commodities while $\mathcal{D}_1^i, \mathcal{D}_2^i$ denote the projections of $\mathcal{D}$ on $\mathbb{R}^{\mathcal{H}_1^i}$ and $\mathbb{R}^{\mathcal{H}_2^i}$. A vector b^i is also defined and used to update $\mathcal{H}_1^i, \mathcal{H}_2^i$. We start with $\mathcal{H}_1^0 = \mathcal{H}$, $\mathcal{H}_2^0 = \emptyset$ and $b^0 = 0$.

If $\alpha_1(\mathcal{D}_1^i) > \gamma\sqrt{k}$ then we consider a traffic vector u^i maximizing $\sum\limits_{h \in \mathcal{H}_1^{i-1}} \frac{d_h}{d_h^{max}}$, otherwise a partition is returned. The vector u^i is then used to update b^i (lines 5-7). Observe that only the components related to commodities inside $\mathcal{H}_1^{i-1}$ are updated while the others do not change. This means that the returned vector $\beta = \sum\limits_{1 \leq i \leq Z} u^i$ (line 19) is such that $\beta \geq b^Z$. The sets $\mathcal{H}_1^i$ and $\mathcal{H}_2^i$ are updated by moving each commodity $h \in \mathcal{H}_1^{i-1}$ to $\mathcal{H}_2^i$ if $b_h^i \geq d_h^{max}$ (lines 8-15). Notice that we always have $\mathcal{H}_1^i \subseteq \mathcal{H}_1^{i-1}$. It is then clear that when the algorithm stops, the obtained partition satisfies what is announced above. The only fact that remains to be proved is that the number of iterations Z is bounded by $2\frac{\sqrt{k}}{\gamma}$.

Algorithm 1 Commodity partitioning algorithm (adapted from [10]).

```
1:  Initialize $i \leftarrow 0$, $\mathcal{H}_1^0 \leftarrow \mathcal{H}, \mathcal{H}_2^0 \leftarrow \emptyset$, $b^0 \leftarrow 0$
2:  while $\alpha_1(\mathcal{D}_1^i) > \gamma\sqrt{k}$ do
3:      $i \leftarrow i + 1$
4:      $u^i \in \arg\max_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}_1^{i-1}} \frac{d_h}{d_h^{max}}$
5:      for all $h \in \mathcal{H}$ do
6:          $b_h^i = \begin{cases} b_h^{i-1} + u_h^i & \text{if } h \in \mathcal{H}_1^{i-1} \\ b_h^{i-1} & \text{otherwise} \end{cases}$
7:      end for
8:      for all $h \in \mathcal{H}_1^{i-1}$ do
9:          if $b_h^i \geq d_h^{max}$ then
10:             $\mathcal{H}_1^i \leftarrow \mathcal{H}_1^{i-1} \backslash \{h\}$
11:             $\mathcal{H}_2^i \leftarrow \mathcal{H}_2^{i-1} \cup \{h\}$
12:         else
13:             $\mathcal{H}_1^i \leftarrow \mathcal{H}_1^{i-1}$, $\mathcal{H}_2^i \leftarrow \mathcal{H}_2^{i-1}$
14:         end if
15:     end for
16: end while
17: $Z \leftarrow i$, $\mathcal{H}_1 \leftarrow \mathcal{H}_1^Z, \mathcal{H}_2 \leftarrow \mathcal{H}_2^Z$
18: $\beta \leftarrow \sum_{1 \leq i \leq Z} u^i$
```

▶ **Lemma 10.** *For any $\gamma > 0$, the commodity set $\mathcal{H}$ can be partitioned in two subsets $\mathcal{H}_1, \mathcal{H}_2$ such that $\alpha_1(\mathcal{D}_1) \leq \gamma\sqrt{k}$ and $\alpha_2(\mathcal{D}_2) \leq \frac{2\sqrt{k}}{\gamma}$ where $\mathcal{D}_1, \mathcal{D}_2$ are the projections of $\mathcal{D}$ on $\mathbb{R}^{\mathcal{H}_1}$ and $\mathbb{R}^{\mathcal{H}_2}$.*

Proof. We only have to prove that $Z \leq \frac{2\sqrt{k}}{\gamma}$. This can be done by slightly modifying the proof of Lemma 10 of [10].

We first argue that $b_h^Z \leq 2d_h^{max}$ for all $h \in \mathcal{H}$. For $h \in \mathcal{H}$, let $i(h)$ be the last iteration number when $h \in \mathcal{H}_1^i$. Therefore we have $b_h^{i(h)-1} \leq d_h^{max}$. Also $u^{i(h)} \leq d_h^{max}$ leading to $b_h^{i(h)} \leq 2d_h^{max}$. Now for $i \geq i(h)$ we have $b_h^Z = b_h^i = b_h^{i(h)}$ implying that, $\sum\limits_{h \in \mathcal{H}} \frac{b_h^Z}{d_h^{max}} \leq 2k$.

Alternatively, $\sum_{h\in\mathcal{H}} \frac{b_h^Z}{d_h^{max}} = \sum_{h\in\mathcal{H}}\sum_{i=1}^{Z} \frac{b_h^i-b_h^{i-1}}{d_h^{max}} = \sum_{i=1}^{Z}\sum_{h\in\mathcal{H}} \frac{b_h^i-b_h^{i-1}}{d_h^{max}} = \sum_{i=1}^{Z}\sum_{h\in\mathcal{H}_1^{i-1}} \frac{u_h^i}{d_h^{max}} \geq \sum_{i=1}^{Z} \gamma\sqrt{k} = Z\gamma\sqrt{k}$. Therefore we have that $Z\gamma\sqrt{k} \leq \sum_{h\in\mathcal{H}} \frac{b_h^Z}{d_h^{max}} \leq 2k$ which implies that $Z \leq \frac{2\sqrt{k}}{\gamma}$. Since β is the sum of Z points in $\mathcal{D}$, we have $\mathcal{B}(\frac{1}{Z}\beta) \subseteq \mathcal{D}$. Moreover, the projection β^2 of β on $\mathbb{R}^{\mathcal{H}_2}$ satisfies $\mathcal{D}_2 \subseteq \mathcal{B}(\beta^2)$ and thus $\alpha_2(\mathcal{D}_2) \leq \frac{2\sqrt{k}}{\gamma}$. ◀

To prove Theorem 4, we only have to take $\gamma = \sqrt{2}$, use Lemma 10, and then invoke Lemma 9 to conclude. Using $k = O(n^2)$, we get that the ratio $\frac{\text{cong}_{\text{sta}}}{\text{cong}_{\text{dyn}}}$ is $O(n)$.

4 Inapproximability with flow restrictions

Let us consider a more general variant of the robust congestion problem where each commodity can only be routed on a subset of allowed edges $E_h \subseteq E$. These restrictions seem to be quite natural to ensure quality of service requirements such as delay constraints.

Observe first that cong_{sta} can still be computed in polynomial-time for this variant. Moreover, the $O(\sqrt{k})$ bound of Section 3 still holds here since all the proofs presented there do not change if we assume that each commodity h can only be routed using edges inside E_h.

The $\Omega(\frac{\log n}{\log\log n})$ inapproximability bound shown for the undirected case [1] (under ETH assumption) still applies to the directed case (with and without flow restrictions). It is however quite far from the $O(\sqrt{k})$ approximation ratio deduced from Section 3. We will prove stronger inapproximability results for the generalisation of cong_{dyn} with flow restrictions under some classical complexity conjectures.

A standard way to prove this kind of results is to first prove that the problem is inapproximable under some constant and then to amplify this constant, see for example [26].

Let us first introduce some additional notations. Taking into account the flow restrictions and given a subset of edges $C \subseteq E$, let $\mathcal{H}_C \subseteq \mathcal{H}$ be the set commodities such that each valid path related to any commodity $h \in \mathcal{H}_C$ intersects C. Even if C is not necessarily a cut in the standard sense of graph theory, C is called a cut in what follows. Given a demand vector $d \in \mathcal{D}$ and a cut C, $\sum_{h\in\mathcal{H}_C} d_h / \sum_{e\in C} c_e$, is obviously a lower bound of $\text{cong}_{\text{dyn}}(\mathcal{D})$. The maximum over all demand vectors $d \in \mathcal{D}$ and all cuts C of the ratio $\sum_{h\in\mathcal{H}_C} d_h / \sum_{e\in C} c_e$ is called cut congestion and denoted by $\text{cong}_{\text{cut}}(\mathcal{D})$. We also use $E_{\mathcal{H}}$ to denote the set of all flow restrictions: $E_{\mathcal{H}} = (E_h)_{h\in\mathcal{H}}$. An instance of cong_{dyn} with flow restrictions is then defined by $(G, c, \mathcal{H}, \mathcal{D}, E_{\mathcal{H}})$.

In Lemma 11, we will prove that it is *NP*-hard to distinguish between instances where $\text{cong}_{\text{dyn}}(\mathcal{D})$ is less than or equal to 1 and those where the cut congestion $\text{cong}_{\text{cut}}(\mathcal{D})$ is greater than or equal to $1+\rho$ for some constant $\rho > 0$. Then, in Lemma 12, we will show that given two instances of this problem, it is possible to build some kind of product instance whose dynamic congestion is less than or equal to the product of the dynamic congestion of the two instances and the cut congestion is greater than or equal to the product of the cut congestion of the two initial instances. Finally, by repetitively using the product of Lemma 12 on the instance of Lemma 11, we can amplify the gap leading to some strong inapproximability results.

Given a 3-SAT instance φ, $\text{val}(\varphi)$ denotes the maximum proportion of clauses that can be simultaneously satisfied (thus φ is satisfiable when $\text{val}(\varphi) = 1$). We will consider polytopes $\mathcal{D}$ that can be expressed through linear constraints and auxiliary variables ξ, i.e., $\mathcal{D} = \{d \in \mathbb{R}^{\mathcal{H}} | Ad + B\xi \leq b\}$ where A and B are matrices of polynomial size (the maximum of the number of columns and the number of rows is polynomially bounded). Notice that it is important to consider polytopes that can be easily described (otherwise the difficulty of solving cong_{dyn} would be a consequence of the difficulty of describing the polytope).

▶ **Lemma 11.** *For $0 < \rho < 1$, there is a polynomial-time mapping from a 3-SAT instance φ to an instance $\mathcal{I} = (G, c, \mathcal{H}, \mathcal{D})$ of $cong_{dyn}$ where $\mathcal{D} = \{d \in \mathbb{R}^{\mathcal{H}} | Ad + B\xi \leq b\}$ such that:*

- *If $val(\varphi) \leq 1 - \rho$ then $cong_{dyn}(\mathcal{I}) \leq 1$*
- *If φ is satisfiable then $cong_{cut}(\mathcal{I}) \geq 1 + \rho$.*

Furthermore, $|V(G)|$, $|E(G)|$, $|\mathcal{H}|$ and the size of the matrices A and B defining $\mathcal{D}$ are all $O(m)$ where m is the number of clauses of φ.

Proof. Given a 3-SAT instance φ with m clauses, we build an instance of cong_{dyn} as follows. We consider two nodes: a source s and a destination t. Then, for each $i = 1, ..., m$ we create a path form s to t containing three directed edges $e_{i,j}$ of capacity 1 for $j = 1, 2, 3$ corresponding to the $i - th$ clause of φ. For each $i = 1, ..., m$ and $j = 1, 2, 3$, $\mathcal{H}$ contains a commodity $h_{i,j}$ with the same source and destination as edge $e_{i,j}$. We also add a commodity $h_{s,t}$ from s to t. The polytope $\mathcal{D}$ is defined as follows. We set $d_{h_{s,t}} = \rho \cdot m$. For each literal l (i.e. a variable or its negation) of the 3-SAT instance φ we add an auxiliary variable ξ_l. Intuitively $\xi_l = 1$ will correspond to setting the literal l to true. For each variable v, we add the constraint $\xi_v + \xi_{\neg v} = 1$ in addition to non-negativity constraints $\xi_v \geq 0$ and $\xi_{\neg v} \geq 0$. For each $i = 1, ..., m$ and $j = 1, 2, 3$, we consider the constraint $d_{h_{i,j}} = \xi_{l_{i,j}}$ where $l_{i,j}$ is the literal appearing in the $i - th$ clause in the $j - th$ position. Observe that the size of $\mathcal{D}$ is $O(m)$. The numbers of nodes, edges and commodities are also $O(m)$.

Consider first the case $\text{val}(\varphi) \leq 1 - \rho$. The set of extreme points of $\mathcal{D}$ is such that the ξ_l variables take their values in $\{0, 1\}$. The maximum dynamic congestion is attained for a demand vector of this form (see Lemma 1). Let d be such a demand vector and consider the corresponding solution of the 3-SAT instance φ. Notice that demand $d_{h_{i,j}}$ can only be routed on $e_{i,j}$. If for some $i = 1, ..., m$ the $i - th$ clause is false, then the demands $d_{h_{i,1}}, d_{h_{i,2}}, d_{h_{i,3}}$ are equal to 0 and therefore one unit of flow of the commodity $h_{s,t}$ can be routed on the path $(e_{i,1}, e_{i,2}, e_{i,3})$. Since $\text{val}(\varphi) \leq 1 - \rho$, there are at least $m \cdot \rho$ such indices i (i.e., false clauses) and therefore the demand $d_{h_{s,t}}$ can be routed with a congestion less than or equal to 1.

We now consider the case where φ is satisfiable. Let d be the demand vector corresponding to a truth assignment satisfying φ. For each $i = 1, ..., m$, let $j(i)$ be the position of a literal set to true in the $i - th$ clause. Therefore we have $d_{h_{i,j(i)}} = 1$ for all $i = 1, ..., m$. Consider the cut $C = \{e_{i,j(i)} | i = 1, ..., m\}$. C intersects the paths related to the m demands $d_{h_{i,j(i)}}$ of value 1 in addition to demand $d_{h_{s,t}}$ of value $m \cdot \rho$. The total capacity of this cut is m while the sum of demands belonging to C is $m + m \cdot \rho$. Therefore the congestion of this cut is $\frac{m + m \cdot \rho}{m} = 1 + \rho$. ◀

Next Lemma (whose proof is provided in Appendix) shows how to build a product instance leading to some gap amplification.

▶ **Lemma 12.** *Given two instances of $cong_{dyn}$ with flow restrictions $\mathcal{I}_1 = (G_1, c_1, \mathcal{H}_1, \mathcal{D}_1, E_{\mathcal{H}_1})$ and $\mathcal{I}_2 = (G_2, c_2, \mathcal{H}_2, \mathcal{D}_2, E_{\mathcal{H}_2})$, we can build a new instance $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 = (G, c, \mathcal{H}, \mathcal{D}, E_{\mathcal{H}})$ such that:*

- *$cong_{dyn}(\mathcal{I}) \leq cong_{dyn}(\mathcal{I}_1) \cdot cong_{dyn}(\mathcal{I}_2)$*
- *$cong_{cut}(\mathcal{I}) \geq cong_{cut}(\mathcal{I}_1) \cdot cong_{cut}(\mathcal{I}_2)$.*

Furthermore, we have $|E(G)| = |E(G_1)| \cdot (|E(G_2)| + 2|V(G_2)|)$, $|V(G)| = |V(G_1)| + |V(G_2)| \cdot |E(G_1)|$, $|\mathcal{H}| = |\mathcal{H}_1| \cdot |\mathcal{H}_2|$ and the size of $\mathcal{D}$ is less than or equal to the product of the sizes of $\mathcal{D}_1$ and $\mathcal{D}_2$.

Combining the two previous lemmas, one can amplify the gap as follows.

▶ **Lemma 13.** *For some $0 < \rho < 1$ and each $p \in \mathbb{N}$, each 3-SAT instance φ can be mapped to an instance $\mathcal{I}^p = (G^p, c^p, \mathcal{H}^p, \mathcal{D}^p, E_{\mathcal{H}^p})$ of $cong_{dyn}$ with flow restrictions where $\mathcal{D}^p = \{d \in \mathbb{R}^{\mathcal{H}^p} | A^p d + B^p \xi \leq b^p\}$ such that:*

- *If $val(\varphi) \leq 1 - \rho$ then $cong_{dyn}(\mathcal{I}^p) \leq 1$*
- *If φ is satisfiable then $cong_{cut}(\mathcal{I}^p) \geq (1 + \rho)^p$.*

Furthermore, there exists a positive constant θ such that $|V(G^p)|$, $|E(G^p)|$, $|\mathcal{H}^p|$ and the size of the matrices A^p and B^p defining $\mathcal{D}^p$ are all less than or equal to $(\theta m)^p$ where m is the number of clauses of φ.

Proof. Let $\mathcal{I}^1$ be the instance defined in Lemma 11. We recursively build $\mathcal{I}^p$ as the product of $\mathcal{I}^{p-1}$ and $\mathcal{I}^1$. Using notation of Lemma 12, we take $\mathcal{I}_1 = \mathcal{I}^{p-1}$, $\mathcal{I}_2 = \mathcal{I}^1$ and $\mathcal{I}^p = \mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2$. Using what is already known about the size of the instance $\mathcal{I}^1$ of Lemma 11 and the results of Lemma 12, a simple induction proves the existence of a constant θ such that $(\theta m)^p$ is an upper bound of the number of vertices, number of edges, number of commodities and the size of the matrices defining the polytope $\mathcal{D}^p$. ◀

By making use of some standard complexity assumptions, inapproxiambility resuls can be directly deduced from the previous lemma.

▶ **Proposition 14.** *Unless $NP \subseteq SUBEXP$, $cong_{dyn}$ with flow restrictions cannot be approximated within a factor of $k^{\frac{c'}{\log\log k}}$ (resp. $n^{\frac{c'}{\log\log n}}$) for some constant c'.*

Proof. *SUBEXP* is the class of problems that can be solved in 2^{n^ϵ} time for all $\epsilon > 0$. Therefore, if $NP \not\subseteq SUBEXP$ then there is a constant $\epsilon_0 > 0$ such that no algorithm can solve the Gap-3-SAT problem in time $O(2^{m^{\epsilon_0}})$ where m is the number of clauses of the 3-SAT instance.
Let $\epsilon_1 < \epsilon_0$ and let $p(m) = \frac{m^{\epsilon_1}}{\log m}$. The size of the instance $\mathcal{I}^{p(m)}$ is polynomial in $m^{p(m)}$. Therefore if we run a polynomial approximation algorithm on the instance $\mathcal{I}^{p(m)}$, the running time will be $m^{c_1 p(m)}$ for some constant c_1. Furthermore, $m^{c_1 p(m)} = m^{c_1 \frac{m^{\epsilon_1}}{\log m}} = 2^{c_1 m^{\epsilon_1}} < 2^{m^{\epsilon_0}}$ for big enough m.

The number of commodities k in the instance $\mathcal{I}^{p(m)}$ is bounded by $(\theta m)^{p(m)}$. We consequently have $\log k \leq \log(\theta m) \frac{m^{\epsilon_1}}{\log m}$ implying that $m > a \log^{\frac{1}{\epsilon_1}} k$ for some constant a and big enough m.

The gap between the congestion of the instances $\mathcal{I}^{p(m)}$ corresponding to a 3-SAT instance for which $val(\varphi) < 1 - \rho$ and those for which $val(\varphi) = 1$ is:

$$(1+\rho)^{p(m)} \quad > (1+\rho)^{p(a \log^{\frac{1}{\epsilon_1}} k)} = (1+\rho)^{\frac{a^{\epsilon_1} \log k}{\frac{1}{\epsilon_1} \log a \log k}} \quad > k^{\frac{c'}{\log\log k}} \quad \text{for some constant c'.}$$

Hence, if a polynomial-time algorithm could solve cong_{dyn} with flow restrictions within an approximation ratio of $O(k^{\frac{c'}{\log\log k}})$, we could use it to solve the Gap-3-SAT problem in $O(2^{m^{\epsilon_0}})$ time. The same proof applies if parameter n (the number of vertices) is considered instead of k. ◀

A slightly weaker inapproximability result is obtained using a weaker complexity assumption (the proof is provided in Appendix).

▶ **Proposition 15.** *Unless $NP \subseteq QP$, $cong_{dyn}$ with flow restrictions cannot be approximated within a factor of $2^{\log^{1-\epsilon} k}$ (resp. $2^{\log^{1-\epsilon} n}$) for any $\epsilon > 0$.*

Using the last part of Proposition 3, all inapproximability results stated for the congestion problem cong_{dyn} are also valid for lin_{dyn}.

References

1 Y. Al-Najjar, W. Ben-Ameur, and J. Leguay. On the approximability of robust network design. *Theoretical Computer Science*, 860:41–50, 2021.

2 D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc. ACM SIGCOMM*, 2003.

3 Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke. Optimal oblivious routing in polynomial time. In *Proc. ACM STOC*, 2003.

4 Y. Bartal and S. Leonardi. Ondashline routing in all-optical networks. *Theoretical Computer Science*, 221(1):19–39, 1999.

5 W. Ben-Ameur. Between fully dynamic routing and robust stable routing. In *International Workshop on Design and Reliable Communication Networks*, 2007.

6 W. Ben-Ameur and H. Kerivin. New economical virtual private networks. *Commun. ACM*, 46(6):69–73, June 2003.

7 W. Ben-Ameur and H. Kerivin. Routing of uncertain traffic demands. *Optimization and Engineering*, 6(3):283–313, 2005.

8 W. Ben-Ameur and M. Żotkiewicz. Robust routing and optimal partitioning of a traffic demand polytope. *International Transactions in Operational Research*, 18(3):307–333, 2011.

9 W. Ben-Ameur and M. Żotkiewicz. Multipolar routing: where dynamic and static routing meet. *Electronic Notes in Discrete Mathematics*, 41:61–68, 2013.

10 D. Bertsimas and V. Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical programming*, 134(2):491–531, 2012.

11 C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–129, September 2007.

12 C. Chekuri, F.B. Shepherd, G. Oriolo, and M.G. Scutellá. Hardness of robust network design. *Networks*, 50(1):50–54, 2007.

13 G. Claßen, A. M. C. A. Koster, M. Kutschka, and I. Tahiri. Robust metric inequalities for network loading under demand uncertainty. *APJOR*, 32(5), 2015.

14 N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive. A flexible model for resource management in virtual private networks. *SIGCOMM Comput. Commun. Rev.*, 29(4):95–108, August 1999.

15 F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella. New approaches for virtual private network design. *SIAM Journal on Computing*, 37(3), 2007.

16 A. Ene, G. Miller, J. Pachocki, and A. Sidford. Routing under balance. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 598–611, New York, NY, USA, 2016. Association for Computing Machinery.

17 J.A. Fingerhut, S. Suri, and J. S. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.

18 A. Frangioni, F. Pascali, and M. G. Scutellà. Static and dynamic routing under disjoint dominant extreme demands. *Operations research letters*, 39(1):36–39, 2011.

19 N. Goyal, N. Olver, and F. Shepherd. Dynamic vs. oblivious routing in network design. In *European Symposium on Algorithms*, pages 277–288. Springer, 2009.

20 N. Goyal, N. Olver, and F. B. Shepherd. The vpn conjecture is true. *Journal of the ACM (JACM)*, 60(3):1–17, 2013.

21 A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proc. ACM STOC*, 2001.

22 A. Gupta and J. Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.

23 M. Hajiaghayi, R. Kleinberg, and T. Leighton. Semi-oblivious routing: lower bounds. In *Proceedings of ACM-SIAM symposium on Discrete algorithms*, pages 929–938, 2007.

24 M. T. Hajiaghayi, R. D. Kleinberg, H. Räcke, and T. Leighton. Oblivious routing on node-capacitated and directed graphs. *ACM Trans. Algorithms*, 3(4), November 2007. doi:10.1145/1290672.1290688.

25 C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *Proc. SPAA*, 2003.

26 I. Haviv and O. Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 469–477, 2007.

27 P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. Lin Lim, and R. Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170. USENIX Association, 2018.

28 H. P. L. Luna. Network planning problems in telecommunications. In *Handbook of Optimization in Telecommunications*, pages 213–240. Springer, 2006.

29 B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. FOCS*, 1997.

30 S. Mattia. The robust network loading problem with dynamic routing. *Comp. Opt. and Appl.*, 54(3):619–643, 2013.

31 M. Minoux. Robust network optimization under polyhedral demand uncertainty is np-hard. *Discrete Applied Mathematics*, 158(5):597–603, 2010.

32 N. Olver. *Robust network design*. PhD thesis, McGill University Library, 2010.

33 N. Olver and F. B. Shepherd. Approximability of robust network design. *Mathematics of Operations Research*, 39(2):561–572, 2014.

34 A. Ouorou and J.P. Vial. A model for robust capacity planning for telecommunications networks under demand uncertainty. In *Workshop on Design and Reliable Communication Networks*, 2007.

35 M. Poss and C. Raack. Affine recourse for the robust network design problem: Between static and dynamic routing. *Networks*, 61(2):180–198, 2013.

36 H. Räcke. Minimizing congestion in general networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 43–52, 2002.

37 H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *40th annual ACM symposium on Theory of Computing*, pages 255–264, 2008.

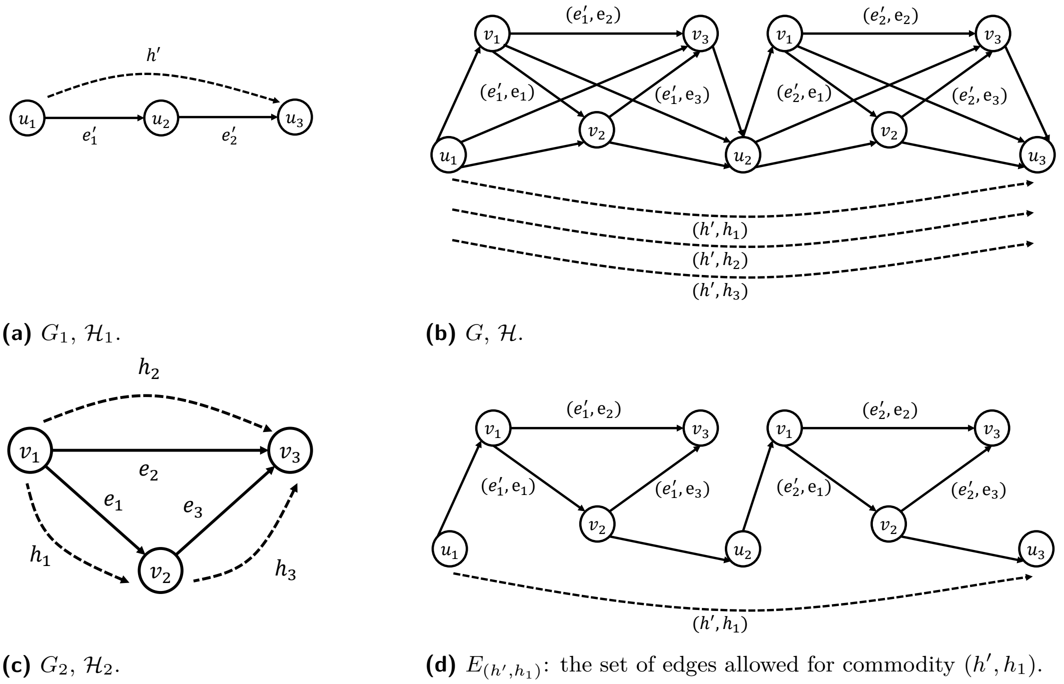
38 N. Wang, Kin H. Ho, G. Pavlou, and M. Howarth. An overview of routing optimization for internet traffic engineering. *IEEE Communications Surveys & Tutorials*, 10(1):36–56, 2008.

39 M. Żotkiewicz and W. Ben-Ameur. More adaptive robust stable routing. In *IEEE Global Telecommunications Conference*, 2009.

A Appendix

Proof of Proposition 8. Let d^* be the demand maximizing $\max_{d\in\mathcal{D}} \sum_{h\in\mathcal{H}} \frac{d_h}{d_h^{max}}$. Since $\mathcal{D}$ is permutation-invariant, $d_h^{max} = d_{h'}^{max}$ for all $h, h' \in \mathcal{H}$ and d^* can be chosen such that $d_h^* = d_{h'}^*$ for all $h, h' \in \mathcal{H}$. Consequently, we have $\alpha_1(\mathcal{D}) \leq k\frac{d_{h_0}^*}{d_{h_0}^{max}}$. Moreover, since $\frac{d_{h_0}^*}{d_{h_0}^{max}}\mathcal{B}(d^{max}) \subseteq \mathcal{D} \subseteq \mathcal{B}(d^{max})$ we also have $\alpha_2(\mathcal{D}) \leq \frac{d_{h_0}^{max}}{d_{h_0}^*}$. Therefore, using notation $x = \frac{d_{h_0}^*}{d_{h_0}^{max}}$, we get that $\min\{\alpha_1(\mathcal{D}), \alpha_2(\mathcal{D})\} \leq \min\{kx, \frac{1}{x}\}$ and x is such that $0 \leq x \leq 1$. To conclude, observe that $\max_{0\leq x\leq 1} \min\{kx, \frac{1}{x}\} = \sqrt{k}$. ◀

Proof of Lemma 12. Let $\mathcal{I}_1 = (G_1, c_1, \mathcal{H}_1, \mathcal{D}_1, E_{\mathcal{H}_1})$ and $\mathcal{I}_2 = (G_2, c_2, \mathcal{H}_2, \mathcal{D}_2, E_{\mathcal{H}_2})$ be two instances of cong_{dyn} with flow restrictions. We denote by G_2' the graph obtained from G_2 by adding two nodes $s(G_2)$ and $t(G_2)$ to G_2, an edge from $s(G_2)$ to each node of G_2 having an infinite capacity (i.e., $|V(G_2)|$ edges), and an edge from each node of G_2 to $t(G_2)$ having also an infinite capacity (i.e., $|V(G_2)|$ edges). We build a graph G by replacing each edge e of G_1 by a copy of G_2' while identifying the node $s(e)$ (resp. $t(e)$) with the node $s(G_2)$ (resp.



(a) G_1, $\mathcal{H}_1$.

(b) G, $\mathcal{H}$.

(c) G_2, $\mathcal{H}_2$.

(d) $E_{(h',h_1)}$: the set of edges allowed for commodity (h', h_1).

Figure 2 Illustration of the construction of the product instance.

$t(G_2))$. Figure 2 illustrates the construction of the product instance. We denote by (e_1, e_2) the edge e_2 in G'_2 corresponding to the copy of G'_2 related to $e_1 \in E(G_1)$. The capacity of the edge (e_1, e_2) is the product of the capacity of edges e_1 and e_2: $c_{(e_1,e_2)} = c_{1e_1} \cdot c_{2e_2}$.

We create a set of commodities $\mathcal{H}$ in G by taking $\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2$ and assuming that $s(h_1, h_2) = s_{h_1}$ and $t(h_1, h_2) = t_{h_1}$ for $(h_1, h_2) \in \mathcal{H}$. We also assume that edges of type $(s(G_2) = s(e), v)$ can only be used by a commodity $(h_1, h_2) \in \mathcal{H}$ if $s(h_2) = v$. Similarly, edges of type $(v, t(G_2) = t(e))$ can only be used by (h_1, h_2) if $t(h_2) = v$. In other words, when a commodity (h_1, h_2) is routed through the copy of G_2 related to an edge $e \in E(G_1)$, then it should enter from $s(h_2)$ and leave at $t(h_2)$ (cf. Figure 2). Other flow restrictions are added by considering the restrictions related to $\mathcal{I}_1$ and $\mathcal{I}_2$. If $h' \in \mathcal{H}_1$ is not allowed to use edge $e' \in E(G_1)$, then all commodities (h', h_2) are not allowed to be routed through the e' copy of G'_2. Moreover, if $e_2 \in E(G_2)$ does not belong to E_{h_2} for some $h_2 \in \mathcal{H}_2$, then for each $e_1 \in E(G_1)$ and each $h_1 \in \mathcal{H}_1$, (e_1, e_2) cannot be used to route commodity (h_1, h_2). Observe that $|E(G)| = |E(G_1)| \cdot (|E(G_2)| + 2|V(G_2)|)$, $|V(G)| = |V(G_1)| + |V(G_2)| \cdot |E(G_1)|$.

We define $\mathcal{D}$ as the set of vectors $d \in \mathbb{R}_+^{\mathcal{H}_1 \times \mathcal{H}_2}$ such that there is a vector $d^1 \in \mathcal{D}_1$ satisfying $d_{h_1,.} \in d^1_{h_1}\mathcal{D}_2$ for all $h_1 \in \mathcal{H}_1$. The constraint $d_{h_1,.} \in d^1_{h_1}\mathcal{D}_2$ can be enforced with linear inequalities as follows. Suppose that $\mathcal{D}_2 = \{d^2 \in \mathbb{R}^{\mathcal{H}_2} | A_2 d^2 + B_2 \xi \leq b_2\}$ for some matrices A_2, B_2. We also assume that this description contains the constraints $d^2_{h_2}/d^{2\,max}_{h_2} \leq 1$ for all $h_2 \in \mathcal{H}_2$ in addition to the non-negativity constraints of demand values $d^2_{h_2}$. Then we can write the constraint $d_{h_1,.} \in d^1_{h_1}\mathcal{D}_2$ as $A_2 d_{h_1,.} + B_2\xi' - d^1_{h_1} b_2 \leq 0$. Indeed, $d^1_{h_1} = 0$ implies $d_{h_1,.} = 0$ while for $d^1_{h_1} > 0$ we have $A_2 d_{h_1,.} + B_2\xi' - d^1_{h_1} b_2 \leq 0$ if and only if $d_{h_1,.}/d^1_{h_1} \in \mathcal{D}_2$. Polytope $\mathcal{D}$ is then defined by constraints $A_2 d_{h_1,.} + B_2 \xi^{h_1} - d^1_{h_1} b_2 \leq 0$ for each $h_1 \in \mathcal{H}_1$ in addition to $A_1 d^1 + B_1 \xi \leq b_1$. Observe that a subscript h_1 is added to express the fact that the auxiliary variables ξ^{h_1} depend on $h_1 \in \mathcal{H}_1$. Notice also that the size of the matrices defining $\mathcal{D}$ is less than or equal to the product of the sizes of the matrices defining $\mathcal{D}_1$ and $\mathcal{D}_2$.

We will now prove that $\mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}) \leq \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_2)$. Let d be a vector in $\mathcal{D}$ and let $d^1 \in \mathcal{D}_1$ be a vector such that $d_{h_1,\cdot} \in d^1_{h_1}\mathcal{D}_2$. For h_1 in $\mathcal{H}_1$, we define $d^{2,h_1} \in \mathcal{D}_2$ by $d^{2,h_1}_{h_2} = \frac{d_{h_1 h_2}}{d^1_{h_1}}$ if $d^1_{h_1} \neq 0$ and $d^{2,h_1} = 0$ if $d^1_{h_1} = 0$. We clearly have $d_{h_1,h_2} = d^1_{h_1} \cdot d^{2,h_1}_{h_2}$ for all $h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2$.

Let f^1, f^{2,h_1} be the optimal routing schemes for $d^1 \in \mathbb{R}^{\mathcal{H}_1}$ and $d^{2,h_1} \in \mathbb{R}^{\mathcal{H}_2}$ for $h_1 \in \mathcal{H}_1$. To route commodity (h_1, h_2), we consider the following multi-commodity flow in G defined by $f_{(h_1,h_2),(e_1,e_2)} = f^1_{h_1,e_1} f^{2,h_1}_{h_2,e_2}$. The total flow on the edge (e_1, e_2) is then given by:

$$\begin{aligned}
\sum_{(h_1,h_2)\in\mathcal{H}_1\times\mathcal{H}_2} d^1_{h_1} d^{2,h_1}_{h_2} f_{(h_1,h_2),(e_1,e_2)} &= \sum_{h_1\in\mathcal{H}_1} d^1_{h_1} f^1_{h_1,e_1} \sum_{h_2\in\mathcal{H}_1} d^{2,h_1}_{h_2} f^{2,h_1}_{h_2,e_2} \\
&\leq \sum_{h_1\in\mathcal{H}_1} d^1_{h_1} f^1_{h_1,e_1} \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_2) c_{2e_2} \\
&\leq \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_2) \cdot c_{1e_1} \cdot c_{2e_2} \\
&= \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_2) \cdot c_{(e_1,e_2)}.
\end{aligned}$$

Since this holds for any edge (e_1, e_2) of G (the other edges of G have an infinite capacity), we deduce that $\mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}) \leq \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{dyn}}(\mathcal{I}_2)$.

Let us now show that $\mathrm{cong}_{\mathrm{cut}}(\mathcal{I}) \geq \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_2)$. Let C_1 (resp. C_2) be a cut of G_1 (resp. G_2) achieving the maximal congestion $\mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_1)$ (resp. $\mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_2)$), and let $d^1 \in \mathcal{D}_1$ (resp. $d^2 \in \mathcal{D}_2$) be a demand vector for which the maximal cut congestion is obtained. In other words, we have $\sum_{h_1\in\mathcal{H}_{C_1}} d^1_{h_1} / \sum_{e_1\in C_1} c_{e_1} = \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_1)$ and $\sum_{h_1\in\mathcal{H}_{C_2}} d^2_{h_2} / \sum_{e_2\in C_2} c_{e_2} = \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_2)$.

Observe that the set of edges $C_1 \times C_2$ is a cut of G that is intersecting all demands of $\mathcal{H}_{C_1} \times \mathcal{H}_{C_2}$. Notice that the flow restrictions that have been considered are crucial here to guarantee the previous fact. Let $d \in \mathbb{R}^{\mathcal{H}}$ be the demand defined by $d_{(h_1,h_2)} = d^1_{h_1} \cdot d^2_{h_2}$. Since $d^1 \in \mathcal{D}_1$ and $d^2 \in \mathcal{D}_2$, we also have $d \in \mathcal{D}$. The congestion on the cut $C_1 \times C_2$ is given by:

$$\begin{aligned}
\frac{\sum_{(h_1,h_2)\in\mathcal{H}_{C_1}\times\mathcal{H}_{C_2}} d_{(h_1,h_2)}}{\sum_{(e_1,e_2)\in C_1\times C_2} c_{(e_1,e_2)}} &= \frac{\sum_{h_1\in\mathcal{H}_{C_1}} d^1_{h_1}}{\sum_{e_1\in C_1} c_{1e_1}} \cdot \frac{\sum_{h_2\in\mathcal{H}_{C_2}} d^2_{h_2}}{\sum_{e_2\in C_2} c_{2e_2}} \\
&= \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_2).
\end{aligned}$$

This clearly implies that $\mathrm{cong}_{\mathrm{cut}}(\mathcal{I}) \geq \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_1) \cdot \mathrm{cong}_{\mathrm{cut}}(\mathcal{I}_2)$. ◀

Proof of Proposition 15. Let us take $p(m) = \log^{c_1}(m)$ for an arbitrary constant c_1. If we run a polynomial-time algorithm on instance the instance $\mathcal{I}^{p(m)}$, we get an algorithm running in poly-logarithmic time. The number of commodities k is bounded by $(\theta m)^{p(m)}$. Thus $\log k \leq \log^{c_1} m \log \theta m < \log^{c_1+2} m$ for big enough m and therefore $m > exp(\log^{\frac{1}{c_1+2}} k)$.

The gap between the congestion of the instances $\mathcal{I}^{p(m)}$ corresponding to 3-SAT instances such that $val(\varphi) < 1 - \rho$ and those such that $val(\varphi) = 1$ is:

$$\begin{aligned}
(1+\rho)^{p(m)} &> (1+\rho)^{p(\log^{\frac{1}{c_1+2}} k))} \\
&= (1+\rho)^{\log^{\frac{c_1}{c_1+2}} k} \\
&> (1+\rho)^{\log^{1-\epsilon} k}
\end{aligned}$$

for any $\epsilon > 0$ if we take c_1 such that $\frac{c_1}{c_1+2} > 1 - \epsilon$. The $(1+\rho)$ term can be replaced by 2 by observing that $2^{\log^{1-\epsilon'} k} = o((1+\rho)^{\log^{1-\epsilon} k})$ for any $\epsilon' < \epsilon$. The same proof applies if parameter n is considered instead of k. ◀

Existential Definability over the Subword Ordering

Pascal Baumann
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Moses Ganardi
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Ramanathan S. Thinniyam
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

We study first-order logic (FO) over the structure consisting of finite words over some alphabet A, together with the (non-contiguous) subword ordering. In terms of decidability of quantifier alternation fragments, this logic is well-understood: If every word is available as a constant, then even the Σ_1 (i.e., existential) fragment is undecidable, already for binary alphabets A.

However, up to now, little is known about the expressiveness of the quantifier alternation fragments: For example, the undecidability proof for the existential fragment relies on Diophantine equations and only shows that recursively enumerable languages over a singleton alphabet (and some auxiliary predicates) are definable.

We show that if $|A| \geq 3$, then a relation is definable in the existential fragment over A with constants if and only if it is recursively enumerable. This implies characterizations for all fragments Σ_i: If $|A| \geq 3$, then a relation is definable in Σ_i if and only if it belongs to the i-th level of the arithmetical hierarchy. In addition, our result yields an analogous complete description of the Σ_i-fragments for $i \geq 2$ of the *pure logic*, where the words of A^* are not available as constants.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic; Theory of computation → Logic

Keywords and phrases subword, subsequence, definability, expressiveness, first order logic, existential fragment, quantifier alternation

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.7

1 Introduction

The subword ordering. A word u is a *subword* of another word v if u can be obtained from v by deleting letters at an arbitrary set of positions. The subword ordering has been studied intensively over the last few decades. On the one hand, it appears in many classical results of theoretical computer science. For example, subwords have been a central topic in string algorithms [4, 12, 28]. Moreover, their combinatorial properties are the basis for verifying lossy channel systems [1]. Particularly in recent years, subwords have received a considerable amount of attention. Notable examples include lower bounds in fine-grained complexity [8, 9] and algorithms to compute the set of all subwords of formal languages [2, 3, 6, 10, 15, 16, 32, 33, 34]. Subwords are also the basis of Simon's congruence [31], which has been studied from algorithmic [13, 14] and combinatorial [5, 11, 20, 22] viewpoints.

First-order logic over subwords. The importance of subwords has motivated the study of first-order logics (FO) over the subword ordering. This has been considered in two variants: In the *pure logic*, one has FO over the structure $(A^*, \preccurlyeq)$, where A is an alphabet and $\preccurlyeq$ is the subword ordering. In the version *with constants*, we have the structure $(A^*, \preccurlyeq, (w)_{w\in A^*})$, which has a constant for each word from A^*. Traditionally for FO, the

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 7; pp. 7:1–7:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

primary questions are *decidability* and *definability*, particularly regarding quantifier alternation fragments Σ_i. Here, decidability refers to the *truth problem*: Given a formula φ in a particular fragment over $(A^*, \preccurlyeq)$ or $(A^*, \preccurlyeq, (w)_{w\in A^*})$, respectively, does φ hold? By definability, we mean understanding which relations can be defined by formulas in a particular fragment. The Σ_i-fragment consists of formulas in prenex form that begin with existential quantifiers and then alternate $i-1$ times between blocks of universal and existential quantifiers. For example, the formula

$$\exists x\colon (a \not\preccurlyeq x \ \vee\ b \not\preccurlyeq x) \ \wedge\ x \not\preccurlyeq u \ \wedge\ x \preccurlyeq v$$

belongs to the Σ_1-fragment, also called the *existential fragment* over $(A^*, \preccurlyeq, (w)_{w\in A^*})$ with $A = \{a, b\}$. The formula has free variables u, v and refers to the constants a and b. It holds if and only if v has more b's or more a's than u.

For FO over subwords, decidability is well-understood. In the pure logic, the Σ_2-fragment is undecidable, already over two letters [17, Corollary III.6], whereas the Σ_1-fragment (i.e., existential formulas) is decidable [25, Theorem 2.2] and NP-complete [21, Theorem 2.1]. This fueled hope that the Σ_1-fragment might even be decidable with constants, but this turned out to be undecidable, already over two letters [17, Theorem III.3]. Decidability (and complexity) have also been studied for the two-variable fragment [21, 22, 26], and extended with counting quantifiers and regular predicates [26, 27].

Nevertheless, little is known about definability. Kudinov, Selivanov, and Yartseva have shown that using arbitrary first-order formulas over $(A^*, \preccurlyeq)$, one can define exactly the relations from the arithmetical hierarchy[1] that are invariant under automorphisms of $(A^*, \preccurlyeq)$ [24, Theorem 5], if $|A| \geq 2$. However, this does not explain definability of the Σ_i-fragments. For example, in order to define all recursively enumerable languages, as far as we can see, their proof requires several quantifier alternations. An undecidability proof by Karandikar and Schnoebelen [21, Theorem 4.6] for the Σ_2-fragment can easily be adapted to show that for each alphabet A, there exists a larger alphabet B such that every recursively enumerable language $L \subseteq A^*$ is definable in the Σ_2-fragment over $(B^*, \preccurlyeq, (w)_{w\in B^*})$. However, a full description of the expressiveness of the Σ_2-fragment is missing.

Existential formulas. The expressiveness of existential formulas is even further from being understood. The undecidability proof in [17] reduces from solvability of Diophantine equations, i.e., polynomial equations over integers, which is a well-known undecidable problem [29]. To this end, it is shown in [17] that the relations $\mathsf{ADD} = \{(a^m, a^n, a^{m+n}) \mid m, n \in \mathbb{N}\}$ and $\mathsf{MULT} = \{(a^m, a^n, a^{m\cdot n}) \mid m, n \in \mathbb{N}\}$ are definable existentially using the subword ordering, if one has at least two letters. Since Diophantine equations can be used to define all recursively enumerable relations over natural numbers, this implies that all recursively enumerable relations involving a single letter are definable existentially. However, this says little about which languages (let alone relations) over more than one letter are definable. For example, it is not clear whether the language of all $w \in \{a, b\}^*$ that do *not* contain aba as an infix, or the reversal relation $\mathsf{REV}_A = \{(u, v) \mid u, v \in A^*,\ v \text{ is the reversal of } u\}$, are definable – it seems particularly difficult to define them over the subword ordering using the methods from [17].

Contribution. We show that for any alphabet A with $|A| \geq 3$, every recursively enumerable relation $R \subseteq (A^*)^k$, $k \in \mathbb{N}$, is existentially definable in $(A^*, \preccurlyeq, (w)_{w\in A^*})$. Since every existentially definable relation is clearly recursively enumerable (via a simple enumerative

[1] Also known as the Kleene–Mostowski hierarchy.

algorithm), this completely describes the expressiveness of existential formulas for $|A| \geq 3$. Despite the undecidability of the existential fragment [17], we find it surprising that all recursively enumerable relations – including relations like REV_A – are existentially definable.

Our result yields characterizations of the Σ_i-fragments for every $i \geq 2$: It implies that for each $i \geq 2$, the Σ_i-fragment over $(A^*, \preccurlyeq, (w)_{w \in A^*})$ can define exactly the relations in Σ_i^0, the i-th level of the arithmetical hierarchy, assuming $|A| \geq 3$. This also provides a description of Σ_i in the pure logic: It follows that in the Σ_i-fragment over $(A^*, \preccurlyeq)$, one can define exactly the relations in Σ_i^0 that are invariant under automorphisms of $(A^*, \preccurlyeq)$, if $|A| \geq 3$.

Since [17] shows that all recursively enumerable languages over one letter are definable in $(A^*, \preccurlyeq, (w)_{w \in A^*})$ if $|A| \geq 2$, it would suffice to define a bijection between a^* and A^* using subwords. However, since this seems hard to do directly, our proof follows a different route. We first show how to define rational transductions and then a special language from which one can build every recursively enumerable relation via rational transductions and intersections. In particular, a byproduct is a direct proof of undecidability of the existential fragment in the case of $|A| \geq 3$ that avoids using undecidability of Diophantine equations[2].

Key ingredients. The undecidability proof for the existential fragment from [17] shows that the relations ADD and MULT are definable, in addition to auxiliary predicates that are needed for this, such as concatenation and letter counting predicates of the form "$|u|_a = |v|_b$". With these methods, it is difficult to express that a certain property holds locally – by which we mean: at every position in a word. Using concatenation, we can define languages like $(a^n b)^*$ for each $n \in \mathbb{N}$ (see Section 3), which "locally look like $a^n b$". But if we want to express that, e.g., aba does not occur as an infix, this is of little help, because words avoiding an infix need not be periodic. The ability to disallow infixes would aid us in defining rational transductions via runs of transducers, as these are little more than configuration sequences where pairs of configurations that are not connected by a transition do not occur as infixes. Such local properties are often easy to state with universal quantification, but this is not available in existential formulas.

An important theme in our proof is to express such local properties by carefully constructing long words in which w has to embed in order for w to have the local property. For example, our first lemma says: Each set $X \subseteq A^{=\ell}$ can be characterized as the set of words (of length $\geq \ell$) that embed into each word in a finite set P. This allows us to define sets X^*.

Steps I–III of our proof use techniques of this type to express rational transductions. In Step IV, we then define the special language $G = \{a^n b^n \mid n \geq 0\}^*$, which has the property that all recursively enumerable languages can be obtained from G using rational transductions and intersection. This yields all recursively enumerable relations over two letters in Step V.

In sum, Steps I–V let us define all recursively enumerable relations over $\{a, b\}$, provided that the alphabet A contains an additional auxiliary letter. It then remains to define recursively enumerable relations that can also involve all other letters in A. We do this in Step VI by observing that each word $w \in A^*$ is determined by its projections to binary alphabets $B \subseteq A$. This allows us to compare words by looking at two letters at a time and use the other (currently unused) letters for auxiliary means.

[2] Our proof relies on the definability of concatenation and certain counting predicates (see Section 3), which was shown directly in [17], without using computational completeness of Diophantine equations.

2 Main results

We say that u is a *subword* of v, written $u \preccurlyeq v$, if there exist words $u_1, \ldots, u_n$ and $v_0, \ldots, v_n$ such that $u = u_1 \cdots u_n$ and $v = v_0 u_1 v_1 \cdots u_n v_n$.

Subword logic. We consider first-order logic over the structure $(A^*, \preccurlyeq)$ and first-order logic over the structure $(A^*, \preccurlyeq, (w)_{w \in A^*})$ enriched with constant symbols w for every word $w \in A^*$. A first-order formula φ with free variables $x_1, \ldots, x_k$ *defines* a relation $R \subseteq (A^*)^k$ if R contains exactly those tuples of words $(w_1, \ldots, w_k)$ that satisfy[3] the formula φ.

Let us define the quantifier alternation fragments of first-order logic. A formula without quantifiers is called Σ_0*-formula* or Π_0*-formula*. For $i \geq 1$, a Σ_i-formula (resp. Π_i-formula) is one of the form $\exists x_1 \cdots \exists x_n \varphi$ (resp. $\forall x_1 \cdots \forall x_n \varphi$), where φ is a a Π_{i-1}-formula (resp. Σ_{i-1}-formula), $x_1, \ldots, x_n$ are variables, and $n \geq 0$. In other words, a Σ_i-formula is in prenex form and its quantifiers begin with a block of existential quantifiers and alternate at most $i-1$ times between universal and existential quantifiers. The Σ_i*-fragment* (Π_i*-fragment*) consists of the Σ_i-formulas (Π_i-formulas). In particular, the Σ_1-fragment (called the *existential fragment*) consists of the formulas in prenex form that only contain existential quantifiers.

Expressiveness with constants. Our main technical contribution is the following.

▶ **Theorem 2.1.** *Let A be an alphabet with $|A| \geq 3$. A relation is definable in the Σ_1-fragment over $(A^*, \preccurlyeq, (w)_{w \in A^*})$ if and only if it is recursively enumerable.*

We prove Theorem 2.1 in Section 3. Theorem 2.1 in particular yields a description of what is expressible using Σ_i-formulas for each $i \geq 1$. Recall that the *arithmetical hierarchy* consists of classes $\Sigma^0_1, \Sigma^0_2, \ldots$, where $\Sigma^0_1 = \mathsf{RE}$ is the class of recursively enumerable relations, and for $i \geq 2$, we have $\Sigma^0_i = \mathsf{RE}^{\Sigma^0_{i-1}}$. Here, for a class of relations $\mathcal{C}$, $\mathsf{RE}^{\mathcal{C}}$ denotes the class of relations recognized by oracle Turing machines with access to oracles over the class $\mathcal{C}$.

▶ **Corollary 2.2.** *Let A be an alphabet with $|A| \geq 3$ and let $i \geq 1$. A relation is definable in the Σ_i-fragment over $(A^*, \preccurlyeq, (w)_{w \in A^*})$ if and only if it belongs to Σ^0_i.*

Expressiveness of the pure logic. Corollary 2.2 completely describes the relations definable in the structure $(A^*, \preccurlyeq, (w)_{w \in A^*})$ if $|A| \geq 3$. We can use this to derive a description of the relations definable without constants, i.e., in the structure $(A^*, \preccurlyeq)$. The lack of constants slightly reduces the expressiveness; to make this precise, we need some terminology. An *automorphism (of $(A^*, \preccurlyeq)$)* is a bijection $\alpha\colon A^* \to A^*$ such that $u \preccurlyeq v$ if and only if $\alpha(u) \preccurlyeq \alpha(v)$. A relation $R \subseteq (A^*)^k$ is *automorphism-invariant* if for every automorphism α, we have $(v_1, \ldots, v_k) \in R$ if and only if $(\alpha(v_1), \ldots, \alpha(v_k)) \in R$. It is straightforward to check that every formula over $(A^*, \preccurlyeq)$ defines an automorphism-invariant relation. Thus, in the Σ_i-fragment over $(A^*, \preccurlyeq)$, we can only define automorphism-invariant relations inside Σ^0_i.

▶ **Corollary 2.3.** *Let A be an alphabet with $|A| \geq 3$ and let $i \geq 2$. A relation is definable in the Σ_i-fragment over $(A^*, \preccurlyeq)$ if and only if it is automorphism-invariant and belongs to Σ^0_i.*

[3] The correspondence between the entries in the tuple and the free variables of φ will always be clear, because the variables will have an obvious linear order by sorting them alphabetically and by their index. For example, if φ has free variables x_i for $1 \leq i \leq k$ and y_j for $1 \leq j \leq \ell$, then we order them as $x_1, \ldots, x_k, y_1, \ldots, y_\ell$.

To give some intuition on automorphism-invariant sets, let us recall the classification of automorphisms of $(A^*, \preccurlyeq)$, shown implicitly by Kudinov, Selivanov, and Yartseva in [24] (for a short and explicit proof, see [18, Lemma 3.8]): A map $\alpha\colon A^* \to A^*$ is an automorphism of $(A^*, \preccurlyeq)$ if and only if (i) the restriction of α to A is a permutation of A, and (ii) α is either a *word morphism*, i.e., $\alpha(a_1 \cdots a_k) = \alpha(a_1) \cdots \alpha(a_k)$ for any $a_1, \ldots, a_k \in A$, or a *word anti-morphism*, i.e., $\alpha(a_1 \cdots a_k) = \alpha(a_k) \cdots \alpha(a_1)$ for any $a_1, \ldots, a_k \in A$.

Finally, Corollary 2.3 raises the question of whether the Σ_1-fragment over $(A^*, \preccurlyeq)$ also expresses exactly the automorphism-invariant recursively enumerable relations. It does not:

▶ **Observation 2.4.** *Let $|A| \geq 2$. There are undecidable relations definable in the Σ_1-fragment over $(A^*, \preccurlyeq)$. However, not every automorphism-invariant regular language is definable in it.*

We prove Corollaries 2.2 and 2.3 and Observation 2.4 in Section 4.

3 Existentially defining recursively enumerable relations

In this section, we prove Theorem 2.1. Therefore, we now concentrate on definability in the Σ_1-fragment. Moreover, for an alphabet A, we will sometimes use the phrase Σ_1*-definable over* A as a shorthand for definability in the Σ_1-fragment over the structure $(A^*, \preccurlyeq, (w)_{w \in A^*})$.

Notation. For an alphabet A, we write $A^{=k}$, $A^{\geq k}$, and $A^{\leq k}$ for the set of words over A that have length exactly k, at least k, and at most k, respectively. We write $|w|$ for the length of a word w. If $B \subseteq A$ is a subalphabet of A then $|w|_B$ denotes the number of occurrences of letters $a \in B$ in w, or simply $|w|_a$ if $B = \{a\}$ is a singleton. Furthermore, we write $\pi_B\colon A^* \to B^*$ for the projection morphism which keeps only the letters from B. If $B = \{a, b\}$, we also write $\pi_{a,b}$ for $\pi_{\{a,b\}}$. The *downward closure* of a word $v \in A^*$ is defined as $v{\downarrow} := \{u \in A^* \mid u \preccurlyeq v\}$.

Basic relations. We will use two kinds of relations, concatenation and counting letters, which are shown to be Σ_1-definable in $(A^*, \preccurlyeq, (w)_{w \in A^*})$ as part of the undecidability proof of the truth problem in [17, Theorem III.3]. The following relations are Σ_1-definable if $|A| \geq 2$.

Concatenation The relation $\{(u, v, w) \in (A^*)^3 \mid w = uv\}$.

Counting letters The relation $\{(u, v) \in (A^*)^2 \mid |u|_a = |v|_b\}$ for any $a, b \in A$.

Moreover, we will make use of a classical fact from word combinatorics: For $u, v \in A^*$, we have $uv = vu$ if and only if there is a word $r \in A^*$ with $u \in r^*$ and $v \in r^*$ [7]. In particular, if p is *primitive*, meaning that $p \in A^+$ and there is no $r \in A^*$ with $|r| < |p|$ and $p \in r^*$, then $up = pu$ is equivalent to $u \in p^*$. Furthermore, note that by counting letters as above, and using concatenation, we can also say $|u|_a = |vw|_a$, i.e., $|u|_a = |v|_a + |w|_a$ for $a \in A$. With these building blocks, we can state arbitrary linear equations over terms $|u|_a$ with $u \in A^*$ and $a \in A$. For example, we can say $|u| = 3 \cdot |v|_a + 2 \cdot |w|_b$ for $u, v, w \in A^*$ and $a, b \in A$. This also allows us to state modulo constraints, such as $\exists v\colon |u|_a = 2 \cdot |v|_a$, i.e., "$|u|_a$ is even". Finally, counting letters lets us define projections: Note that for $B \subseteq A$ and $u, v \in A^*$, we have $v = \pi_B(u)$ if and only if $v \preccurlyeq u$ and $|v|_b = |u|_b$ for each $b \in B$ as well as $\neg(a \preccurlyeq v)$ for every $a \in A \setminus B$.

For any subalphabet $B \subseteq A$ one can clearly define B^* over A. Hence definability of a relation over B also implies definability over the larger alphabet A.

Finite state transducers. An important ingredient of our proof is to define regular languages in the subword order, and, more generally, rational transductions, i.e., relations recognized by finite state transducers.

For $k \in \mathbb{N}$, a *k-ary finite state transducer* $\mathcal{T} = (Q, A, \delta, q_0, Q_f)$ consists of a finite set of *states* Q, an input alphabet A, an *initial state* $q_0 \in Q$, a set of *final states* $Q_f \subseteq Q$, and a *transition relation* $\delta \subseteq Q \times (A \cup \{\varepsilon\})^k \times Q$. For a *transition* $(q, a_1, \dots, a_k, q') \in \delta$, we also write $q \xrightarrow{(a_1,\dots,a_k)} q'$.

The transducer $\mathcal{T}$ *recognizes* the k-ary relation $R(\mathcal{T}) \subseteq (A^*)^k$ containing precisely those k-tuples $(w_1, \dots, w_k)$, for which there is a transition sequence $q_0 \xrightarrow{(a_{1,1},\dots,a_{k,1})} q_1 \xrightarrow{(a_{1,2},\dots,a_{k,2})} \dots \xrightarrow{(a_{1,m},\dots,a_{k,m})} q_m$ with $q_m \in Q_f$ and $w_i = a_{i,1}a_{i,2}\cdots a_{i,m}$ for all $i \in \{1, \dots, k\}$. Such a transition sequence is called an *accepting run* of $\mathcal{T}$. We sometimes prefer to think of the w_i as *produced output* rather than *consumed input* and thus occasionally use terminology accordingly. A relation T is called a *rational transduction* if it is recognized by some finite state transducer $\mathcal{T}$. Unary transducers (i.e., $k = 1$) recognize the *regular languages.*

Overview. As outlined in the introduction, our proof consists of six steps. In Steps I–III, we show that we can define all rational transductions $T \subseteq (A^*)^k$ over the alphabet B, if $|B| \geq |A| + 1$. In Step IV, we define the special language $G = \{a^n b^n \mid n \geq 0\}^*$. From G, all recursively enumerable languages can be obtained using rational transductions and intersection, which in Step V allows us to define over B all recursively enumerable relations over A, provided that $|B| \geq |A| + 1$. Finally, in Step VI, we use projections to binary alphabets to define arbitrary recursively enumerable relations over A, if $|A| \geq 3$.

Step I: Defining Kleene stars. We first define the languages X^*, where X consists of words of equal length. To this end, we establish an alternative representation for such sets.

▶ **Lemma 3.1.** *Every nonempty set $X \subseteq A^{=\ell}$ can be written as $X = A^{\geq \ell} \cap \bigcap_{p \in P} p{\downarrow}$ for some finite set $P \subseteq A^*$.*

Proof. We can assume $\ell \geq 1$ since otherwise $X = \{\varepsilon\} = A^{\geq 0} \cap \varepsilon{\downarrow}$. Let $w \in A^*$ be any permutation of A (i.e., each letter of A appears exactly once in w). If $a \in A$, then $(w \setminus a)$ denotes the word obtained from w by deleting a. For any nonempty word $u = a_1 \cdots a_k \in A^+$, $a_1, \dots, a_k \in A$, define the word

$$p_u = (w \setminus a_1)\,(w \setminus a_1)\,a_1\,(w \setminus a_2)\,(w \setminus a_2)\,a_2 \cdots (w \setminus a_{k-1})\,(w \setminus a_{k-1})\,a_{k-1}\,(w \setminus a_k)\,(w \setminus a_k).$$

Note that p_u does not contain u as a subword: In trying to embed each letter a_i of u into p_u, the first possible choice for a_1 comes after the initial sequence $(w \setminus a_1)\,(w \setminus a_1)$. Similarly, the next possible choice for each subsequent a_i is right after $(w \setminus a_i)\,(w \setminus a_i)$. However, this only works until a_{k-1}, since there is no a_k at the end of p_u.

On the other hand, observe that p_u contains every word $v \in A^{\leq k} \setminus \{u\}$ as a subword: Suppose that $v = b_1 \cdots b_m$, $b_1, \dots, b_m \in A$, and let $i \in [1, m+1]$ be the minimal position with $b_i \neq a_i$ or $i = m + 1$. The prefix $b_1 \cdots b_{i-1} = a_1 \cdots a_{i-1}$ occurs as a subword of p_u, which in the case $i = m + 1$ already is the whole word v. If $i \leq m$ then b_i occurs in $(w \setminus a_i)$, and $b_{i+1} \cdots b_m$ embeds into the subword $(w \setminus a_i)\,a_i \cdots (w \setminus a_{k-1})\,a_{k-1}$ of p_u. Thus, we can write

$$X \;= A^{\geq \ell} \,\cap \bigcap_{u \in (A^{=\ell} \setminus X) \cup A^{=\ell+1}} p_u{\downarrow}.$$

Here $u \in A^{=\ell+1}$ was added to also exclude all words of length greater than ℓ. ◀

▶ **Lemma 3.2.** *Let $A \subseteq B$ be finite alphabets and $\# \in B \setminus A$. Let $X \subseteq A^{=k}$ and $Y \subseteq A^{=\ell}$ be sets. Then $(X\#Y\#)^*$ and X^* are Σ_1-definable over B.*

Proof. We can clearly assume that X, Y are nonempty. By Lemma 3.1 we can write $X = A^{\geq k} \cap \bigcap_{p \in P} p{\downarrow}$ and $Y = A^{\geq \ell} \cap \bigcap_{q \in Q} q{\downarrow}$ for some finite sets $P, Q \subseteq A^*$. We claim that $w \in (A \cup \{\#\})^*$ belongs to $(X\#Y\#)^*$ if and only if

$$\exists n \in \mathbb{N} \colon |w|_\# = 2n \wedge |w|_A = (k+\ell) \cdot n \wedge \bigwedge_{p \in P, q \in Q} w \preccurlyeq (p\#q\#)^n. \tag{1}$$

Observe that the number n is uniquely determined by $|w|_\#$. The “only if”-direction is clear. Conversely, suppose that $w \in (A \cup \{\#\})^*$ satisfies the formula. We can factorize $w = x_1\#y_1\# \dots x_n\#y_n\#$ where each x_i is a subword of each word $p \in P$, and each y_i is a subword of each word $q \in Q$. If some word x_i were strictly longer than k, then it would belong to X by the representation of X, and in particular would have length k, contradiction. Therefore each word x_i has length at most k, and similarly each word y_i has length at most ℓ. However, since the total length of $x_1 y_1 \dots x_n y_n$ is $(k+\ell) \cdot n$, we must have $|x_i| = k$ and $|y_i| = \ell$, and hence $x_i \in X$ and $y_i \in Y$ for all $i \in [1, n]$. This proves our claim.

Finally, (1) is equivalent to the following Σ_1-formula:

$$(k+\ell) \cdot |w|_\# = 2 \cdot |w|_A \;\wedge \bigwedge_{p \in P, q \in Q} \exists u \in (p\#q\#)^* \colon (w \preccurlyeq u \;\wedge\; |u|_\# = |w|_\#)$$

Here, we express $u \in (p\#q\#)^*$ as follows. If $p \neq q$, then $p\#q\#$ is primitive and $u \in (p\#q\#)^*$ is equivalent to $u(p\#q\#) = (p\#q\#)u$. If $p = q$, then $u \in (p\#q\#)^*$ is equivalent to $up\# = p\#u$ and $|u|_\#$ being even. Finally, to define X^* we set $Y = \{\varepsilon\}$ and obtain $X^* = \pi_A((X\#Y\#)^*)$. ◀

Step II: Blockwise transductions. On our way towards rational transductions, we work with a subclass of transductions. If $T \subseteq A^* \times A^*$ is any subset, then we define the relation

$$T^* = \{(x_1 \cdots x_n,\, y_1 \cdots y_n) \mid n \in \mathbb{N},\; (x_1, y_1), \dots, (x_n, y_n) \in T\}.$$

We call a transduction *blockwise* if it is of the form T^* for some $T \subseteq A^{=k} \times A^{=\ell}$ and $k, \ell \in \mathbb{N}$.

▶ **Lemma 3.3.** *Let $A \subseteq B$ be finite alphabets with $|B| \geq |A|+1$. Every blockwise transduction $R \subseteq A^* \times A^*$ is Σ_1-definable over B.*

Proof. Let $\# \in B \setminus A$ be a symbol. Suppose that $R = T^*$ for some $T \subseteq A^{=k} \times A^{=\ell}$. Define the language $L = \{x\#y\# \mid (x, y) \in T\}^*$. Note that

$$w \in L \iff w \in (A^{=k}\#A^{=\ell}\#)^* \;\wedge\; \pi_A(w) \in \{xy \mid (x, y) \in T\}^*,$$

and hence L is Σ_1-definable over B by Lemma 3.2. The languages $X = (A^{=k}\#\#)^*$ and $Y = (\#A^{=\ell}\#)^*$ are also definable over B by Lemma 3.2. Then $(x, y) \in R$ if and only if

$$\exists w \in L,\, \hat{x} \in X,\, \hat{y} \in Y \colon\; \hat{x}, \hat{y} \preccurlyeq w \;\wedge\; |w|_\# = |\hat{x}|_\# = |\hat{y}|_\# \;\wedge\; x = \pi_A(\hat{x}) \;\wedge\; y = \pi_A(\hat{y}).$$ ◀

Step III: Rational transductions. We are ready to define arbitrary rational transductions.

▶ **Lemma 3.4.** *Let $A \subseteq B$ be finite alphabets where $|A|+1 \leq |B|$ and $|B| \geq 3$. Every rational transduction $T \subseteq (A^*)^k$ is Σ_1-definable over B.*

Proof. Let $a, b \in B$. Let us first give an overview. Suppose the transducer for T has n transitions. Of course, we may assume that every run contains at least one transition. The idea is that a sequence of transitions is encoded by a word, where transition $j \in \{1, \ldots, n\}$ is represented by $a^j b^{n+1-j}$. We will define predicates run and input_i for $i \in \{1, \ldots, k\}$ with

$$(w_1, \ldots, w_k) \in T \iff \exists w \in \{a, b\}^* \colon\ \mathsf{run}(w) \ \wedge\ \bigwedge_{i=1}^{k} \mathsf{input}_i(w, w_i).$$

Here, $\mathsf{run}(w)$ states that w encodes a sequence of transitions that is a run of the transducer. Moreover, $\mathsf{input}_i(w, w_i)$ states that $w_i \in A^*$ is the input of this run in the i-th coordinate.

We begin with the predicate run. Let us call the words in $X = \{a^j b^{n+1-j} \mid j \in \{1, \ldots, n\}\}$ the *transition codes*. Let Δ be the set of all words $a^i b^{n+1-i} a^j b^{n+1-j}$ for which the target state of transition i and the source state of transition j are the same. Note that a word $w \in X^*$ represents a run if

1. w begins with a transition that can be applied in an initial state,
2. w ends with a transition that leads to a final state, and
3. either $w \in \Delta^* \cap X\Delta^* X$ or $w \in X\Delta^* \cap \Delta^* X$, depending on whether the run has an even or an odd number of transitions.

Thus, we can define $\mathsf{run}(w)$ using prefix and suffix relations and membership to sets Δ^*. The prefix and suffix relation can be defined over $\{a, b\}$. Finally, we can express $w \in X^*$, $w \in \Delta^*$ and similar with Lemma 3.2.

It remains to define the input_i predicate. In the case that every transition reads a single letter on each input (i.e., no ε input), we can simply replace each transition code in w by its i-th input letter using a blockwise transduction. To handle ε inputs, we define input_i in two steps. Fix i and let $A = \{a_1, \ldots, a_m\}$. We first obtain an encoded version u_i of the i-th input from w: For every transition that reads a_j, we replace its transition code with $ab^j ab^{m-j} a$. Moreover, for each transition that reads ε, we replace the transition code by b^{m+3}. Using Lemma 3.3, this replacement is easily achieved using a blockwise transduction. Hence, each possible input in $A \cup \{\varepsilon\}$ is encoded using a block from $Y \cup \{b^{m+3}\}$, where $Y = \{ab^j ab^{m-j} a \mid j \in \{1, \ldots, m\}\}$.

Suppose we have produced the encoded input $u_i \in (Y \cup \{b^{m+3}\})^*$. In the next step, we want to define the word $v_i \in Y^*$, which is obtained from u_i by removing each block b^{m+3} from u_i. We do this as follows:

$$v_i \in Y^* \ \wedge\ v_i \preccurlyeq u_i \ \wedge\ |v_i|_a = |u_i|_a.$$

Note that here, we can express $v_i \in Y^*$ because of Lemma 3.2. In the final step, we turn v_i into the input $w_i \in A$ by replacing each block $ab^j ab^{m-j} a$ with a_j for $j \in \{1, \ldots, m\}$. This is just a blockwise transduction and can be defined by Lemma 3.3 because $|B| \geq |A| + 1$. ◀

▶ Remark 3.5. We do not use this here, but Lemma 3.4 also holds without the assumption $|B| \geq 3$. Indeed, if $|B| = 2$, then this would imply $|A_i| = 1$ for every i. Then we can write $A_i = \{a_i\}$ for (not necessarily distinct) letters $a_1, \ldots, a_k$. Since T is rational, the set of all $(x_1, \ldots, x_k) \in \mathbb{N}^k$ with $(a_1^{x_1}, \ldots, a_k^{x_k}) \in T$ is semilinear, and thus Σ_1-definable in $(\mathbb{N}, +, 0)$. It follows from the known predicates that T is Σ_1-definable using subwords over $\{a_1, \ldots, a_k\}$.

Step IV: Generator language. Our next ingredient is to express a particular non-regular language G (and its variant $G_\#$):

$$G = \{a^n b^n \mid n \geq 0\}^*, \qquad G_\# = \{a^n b^n \# \mid n \geq 0\}^*.$$

This will be useful because from G, one can produce all recursively enumerable sets by way of rational transductions and intersection.

▶ **Lemma 3.6.** *The language $\{ab, \#\}^*$ is Σ_1-definable over $\{a, b, \#\}$.*

Proof. Note that

$$u \in \{ab, \#\}^* \iff \exists v \in \#^*ab\#^*,\ w \in v^*\colon\ u \preccurlyeq w \ \wedge\ \pi_{a,b}(u) = \pi_{a,b}(w).$$

Here, the language $\#^*ab\#^*$ can be defined using concatenation. Moreover, since every word in $\#^*ab\#^*$ is primitive, we express $w \in v^*$ by saying $vw = wv$. ◀

▶ **Lemma 3.7.** *Let $\{a, b\} \subseteq A$ and $|A| \geq 3$. The language G is Σ_1-definable over A.*

Proof. Suppose $\# \in A \setminus \{a, b\}$. Since $G = \pi_{a,b}(G_\#)$, it suffices to define $G_\#$. We can define the language $a^*b^*\#$ as a concatenation of a^*, b^*, and $\#$. The next step is to define the language $K = (a^*b^*\#)^*$. To this end, notice that

$$w \in K \iff \exists u \in a^*b^*\#,\ v \in u^*\colon\ w \preccurlyeq v \ \wedge\ |w|_\# = |v|_\#.$$

Here, since the words in $a^*b^*\#$ are primitive, we can express $v \in u^*$ by saying $vu = uv$. Thus, we can define K. Using K and Lemma 3.6, we can define $G_\#$, since

$$w \in G_\# \iff w \in K \ \wedge\ \exists v \in \{ab, \#\}^*\colon\ \pi_{a,\#}(w) = \pi_{a,\#}(v) \ \wedge\ \pi_{b,\#}(w) = \pi_{b,\#}(v). \quad ◀$$

Step V: Recursively enumerable relations over two letters. We are now ready to define all recursively enumerable relations over two letters in $(A^*, \preccurlyeq, (w)_{w\in A^*})$, provided that $|A| \geq 3$. For two rational transductions $T \subseteq A^* \times B^*$ and $S \subseteq B^* \times C^*$, and a language $L \subseteq A^*$, we denote *application* of T to L as $TL = \{v \in B^* \mid \exists u \in L\colon (u, v) \in T\} \subseteq B^*$, and we denote *composition* of S and T as $S \circ T = \{(u, w) \mid \exists v \in B^*\colon (u, v) \in T \wedge (v, w) \in S\} \subseteq A^* \times C^*$. The latter is again a rational transduction (see e.g. [7]).

▶ **Lemma 3.8** (Hartmanis & Hopcroft 1970). *Every recursively enumerable language L can be written as $L = \alpha(T_1G_\# \cap T_2G_\#)$ with a morphism α and rational transductions T_1, T_2.*

Proof. This follows directly from [19, Theorem 1] and the proof of [19, Theorem 2]. ◀

Let us briefly sketch the proof of Lemma 3.8. It essentially states that every recursively enumerable language can be accepted by a machine with access to two counters that work in a restricted way. The two counters have instructions to *increment*, *decrement*, and *zero test* (which correspond to the letters a, b, and $\#$ in $G_\#$). The restriction, which we call "locally one-reversal" (L1R) is that in between two zero tests of some counter, the instructions of that counter must be *one-reversal*: There is a phase of increments and then a phase of decrements (in other words, after a decrement, no increments are allowed until the next zero test).

To show this, Hartmanis and Hopcroft use the classical fact that every recursively enumerable language can be accepted by a four counter machine (without the L1R property). Then, the four counter values p, q, r, s can be encoded as $2^p3^q5^r7^s$ in a single integer register that can (i) multiply with, (ii) divide by, (iii) test non-divisibility by the constants $2, 3, 5, 7$. Such a register, in turn, is easily simulated using two L1R-counters: For example, to multiply by $f \in \{2, 3, 5, 7\}$, one uses a loop that decrements the first counter and increments the second by f, until the first counter is zero. The other instructions are similar.

▶ **Lemma 3.9.** *For every recursively enumerable relation $R \subseteq (\{a, b\}^*)^k$, there is a rational transduction $T \subseteq (\{a, b\}^*)^{k+2}$ such that*

$$(w_1, \ldots, w_k) \in R \iff \exists u, v \in G\colon (w_1, \ldots, w_k, u, v) \in T. \tag{2}$$

Proof. We shall build T out of several other transductions. These will be over larger alphabets, but since we merely compose them to obtain T, this is not an issue.

A standard fact from computability theory states that a relation is recursively enumerable if and only if it is the homomorphic image of some recursively enumerable language. In particular, there is a recursively enumerable language $L \subseteq B^*$ and morphisms $\beta_1, \ldots, \beta_k$ such that $R = \{(\beta_1(w), \ldots, \beta_k(w)) \mid w \in L\}$. By Lemma 3.8, we may write $L = \alpha(T_1 G_\# \cap T_2 G_\#)$ for a morphism $\alpha\colon C^* \to B^*$ and rational transductions $T_1, T_2 \subseteq \{a, b, \#\}^* \times C^*$.

Notice that if $\gamma\colon \{a, b, \#\}^* \to \{a, b\}^*$ is the morphism with $\gamma(a) = a$, $\gamma(b) = b$, and $\gamma(\#) = abab$, then $G_\# = (a^*b^*\#)^* \cap \gamma^{-1}(G)$. This means, there is a rational transduction $S \subseteq \{a, b\}^* \times \{a, b, \#\}^*$ with $G_\# = SG$. Therefore, we can replace $G_\#$ in the above expression for L and arrive at $L = \alpha\big((T_1 \circ S)G \cap (T_2 \circ S)G\big)$. In sum, we observe that $(w_1, \ldots, w_k) \in R$ if and only if there exists a $w \in C^*$ with $w \in (T_1 \circ S)G$ and $w \in (T_2 \circ S)G$ such that $w_i = \beta_i(\alpha(w))$ for $i \in \{1, \ldots, k\}$. Consider the relation

$$T = \{(\beta_1(\alpha(w)), \ldots, \beta_k(\alpha(w)), u, v) \mid w \in C^*,\ (u, w) \in T_1 \circ S,\ (v, w) \in T_2 \circ S\}.$$

Note that T is rational: A transducer can guess w, letter by letter, and on track $i \in \{1, \ldots, k\}$, it outputs the image under $\beta_i(\alpha(\cdot))$ of each letter. To compute the output on tracks $k+1$ and $k+2$, it simulates transducers for $T_1 \circ S$ and $T_2 \circ S$. Moreover, we have $T \subseteq (\{a, b\}^*)^{k+2}$ and our observation implies that (2) holds. ◀

▶ **Lemma 3.10.** *Let A be an alphabet with $\{a, b\} \subseteq A$ and $|A| \geq 3$. Then every recursively enumerable relation $R \subseteq (\{a, b\}^*)^k$ is Σ_1-definable over A.*

Proof. Take the rational transduction T as in Lemma 3.9. Since $T \subseteq (\{a, b\}^*)^{k+2}$ and $|A| \geq |\{a, b\}| + 1$, Lemma 3.4 and Lemma 3.7 yield the result. ◀

Step VI: Arbitrary recursively enumerable relations. We have seen that if $|A| \geq 3$, then we can define over A every recursively enumerable relation over two letters. In the proof, we use a third letter as an auxiliary letter. Our last step is to define all recursively enumerable relations that can use all letters of A freely. This clearly implies Theorem 2.1. To this end, we observe that every word is determined by its binary projections.

▶ **Lemma 3.11.** *Let A be an alphabet with $|A| \geq 2$ and let $u, v \in A^*$ such that for every binary alphabet $B \subseteq A$, we have $\pi_B(u) = \pi_B(v)$. Then $u = v$.*

Proof. Towards a contradiction, suppose $u \neq v$. We clearly have $|u| = |v|$. Thus, if $w \in A^*$ is the longest common prefix of u and v, then $u = wau'$ and $v = wbv'$ for some letters $a \neq b$ and words $u', v' \in A^*$. But then the words $\pi_{a,b}(u)$ and $\pi_{a,b}(v)$ differ: After the common prefix $\pi_{a,b}(w)$, the word $\pi_{a,b}(u)$ continues with a and the word $\pi_{a,b}(v)$ continues with b. ◀

We now fix $a, b \in A$ with $a \neq b$. For any binary alphabet $B \subseteq A$ let $\rho_B\colon A^* \to \{a, b\}^*$ be any morphism with $\rho_B(B) = \{a, b\}$ and $\rho_B(c) = \varepsilon$ for all $c \in A \setminus B$, i.e., ρ_B first projects a word over A to B and then renames the letters from B to $\{a, b\}$. Recall that $\binom{|A|}{2}$ is the number of binary alphabets $B \subseteq A$. We define the encoding function $e\colon A^* \to (\{a, b\}^*)^{\binom{|A|}{2}}$ which maps a word $u \in A^*$ to the tuple consisting of all words $\rho_B(u)$ for all binary alphabets $B \subseteq A$ (in some arbitrary order). Note that e is injective by Lemma 3.11.

▶ **Lemma 3.12.** *If $|A| \geq 3$, then $e\colon A^* \to (\{a, b\}^*)^{\binom{|A|}{2}}$ is Σ_1-definable over A.*

Proof. For binary alphabets $B, C \subseteq A$, a map $\sigma\colon B^* \to C^*$ is called a *binary renaming* if (i) σ is a word morphism and (ii) σ restricted to B is a bijection of B and C. If, in addition, there is a letter $\# \in B \cap C$ such that $\sigma(\#) = \#$, then we say that σ *fixes a letter*.

Observe that if we can Σ_1-define all binary renamings, then the encoding function e can be Σ_1-defined using projections and binary renamings. Thus, it remains to define all binary renamings. For this, note that every binary renaming can be written as a composition of (at most three) binary renamings that each fix some letter. Hence, it suffices to define any binary renaming that fixes a letter. Suppose $\sigma\colon \{c,\#\}^* \to \{d,\#\}^*$ with $\sigma(c) = d$ and $\sigma(\#) = \#$. Without loss of generality, we assume $c \neq d$. Then σ is Σ_1-definable since

$$\sigma(u) = v \iff \exists w \in \{cd,\#\}^*\colon\ u = \pi_{c,\#}(w) \ \wedge\ v = \pi_{d,\#}(w).$$

and $\{cd,\#\}^*$ is definable by Lemma 3.6. ◀

▶ **Theorem 3.13.** *Let A be an alphabet with $|A| \geq 3$. Then every recursively enumerable relation $R \subseteq (A^*)^k$ is Σ_1-definable in $(A^*, \preccurlyeq, (w)_{w\in A^*})$.*

Proof. The encoding function e is clearly computable and injective by Lemma 3.11. Therefore a relation $R \subseteq (A^*)^k$ is recursively enumerable if and only if the image

$$e(R) = \{(e(w_1),\dots,e(w_k)) \mid (w_1,\dots,w_k) \in R\} \subseteq (\{a,b\}^*)^{k\cdot\binom{|A|}{2}}$$

is recursively enumerable. This means that $e(R)$ is Σ_1-definable over A by Lemma 3.10. Thus, we can define R as well, since we have

$$(w_1,\dots,w_k) \in R \iff (e(w_1),\dots,e(w_k)) \in e(R),$$

and the function e is Σ_1-definable over A by Lemma 3.12. ◀

4 Consequences for other fragments

In this section, we prove Corollaries 2.2 and 2.3 and Observation 2.4. When working with higher levels (Σ_i^0 for $i \geq 2$) of the arithmetic hierarchy, it will be convenient to use a slightly different definition than the one using oracle Turing machines: [23, Theorem 35.1] implies that for $i \geq 1$, a relation $R \subseteq (A^*)^k$ belongs to Σ_{i+1}^0 if and only if it can be written as $R = \pi((A^*)^{k+\ell} \setminus S)$, where $S \subseteq (A^*)^{k+\ell}$ is a relation in Σ_i^0 and $\pi\colon (A^*)^{k+\ell} \to (A^*)^k$ is the projection to the first k coordinates.

Proof of Corollary 2.2. It is immediate that every predicate definable in the Σ_i-fragment of $(A^*, \preccurlyeq, (w)_{w\in A^*})$ belongs to Σ_i^0, because the subword relation is recursively enumerable. We show the converse using induction on i, such that Theorem 2.1 is the base case.

Now suppose that every relation in Σ_i^0 is definable in the Σ_i-fragment of $(A^*, \preccurlyeq, (w)_{w\in A^*})$ and consider a relation $R \subseteq (A^*)^k$ in Σ_{i+1}^0. Then we can write $R = \pi((A^*)^{k+\ell} \setminus S)$ for some $\ell \geq 0$, where $\pi\colon (A^*)^{k+\ell} \to (A^*)^k$ is the projection to the first k coordinates, and $S \subseteq (A^*)^{k+\ell}$ is a relation in Σ_i^0. By induction, S is definable by a Σ_i-formula φ over $(A^*, \preccurlyeq, (w)_{w\in A^*})$. By negating φ and moving all negations inwards, we obtain a Π_i-formula ψ that defines $(A^*)^{k+\ell} \setminus S$. Finally, adding existential quantifiers for the variables corresponding to the last ℓ coordinates yields a Σ_{i+1}-formula for $R = \pi((A^*)^{k+\ell} \setminus S)$. ◀

Proof of Corollary 2.3. Clearly, every relation definable with a Σ_i-formula over $(A^*, \preccurlyeq)$ must be automorphism-invariant and must define a relation in Σ_i^0.

Conversely, consider an automorphism-invariant relation $R \subseteq (A^*)^k$ in Σ_i^0. Then R is definable using a Σ_i-formula φ with free variables $x_1, \ldots, x_k$ over $(A^*, \preccurlyeq, (w)_{w\in A^*})$ by Corollary 2.2. Let $w_1, \ldots, w_\ell$ be the constants occurring in φ. From φ, we construct the Σ_i-formula φ' over $(A^*, \preccurlyeq)$, by replacing each occurrence of w_j by a fresh variable y_j.

It was shown in [21, Sections 4.1 and 4.2] that from the tuple $(w_1, \ldots, w_\ell) \in (A^*)^\ell$, one can construct a Σ_2-formula ψ with free variables $y_1, \ldots, y_\ell$ over $(A^*, \preccurlyeq)$ such that $\psi(u_1, \ldots, u_\ell)$ is true if and only if there exists an automorphism of $(A^*, \preccurlyeq)$ mapping u_j to w_j for each j. We claim that the formula $\chi = \exists y_1, \ldots, y_\ell \colon \psi \wedge \varphi'$ defines the set R. Since ψ belongs to Σ_2 and thus χ belongs to Σ_i, this implies the corollary.

Clearly, every $(v_1, \ldots, v_k) \in R$ satisfies χ. Moreover, if $\chi(v_1, \ldots, v_k)$, then there are $u_1, \ldots, u_\ell \in A^*$ with $\varphi'(v_1, \ldots, v_k, u_1, \ldots, u_\ell)$ and an automorphism α mapping u_j to w_j for each j. Since α is an automorphism, the formula φ' is also satisfied on the tuple $(\alpha(v_1), \ldots, \alpha(v_k), \alpha(u_1), \ldots, \alpha(u_\ell)) = (\alpha(v_1), \ldots, \alpha(v_k), w_1, \ldots, w_\ell)$ and thus we have $(\alpha(v_1), \ldots, \alpha(v_k)) \in R$. Since R is automorphism-invariant, this implies $(v_1, \ldots, v_k) \in R$. ◀

Proof of Observation 2.4. Take a recursively enumerable, but undecidable subset $S \subseteq \mathbb{N}$. Fix a letter $a \in A$ and define the unary language $L = \{a^n \mid n \in S\}$, which is definable by a Σ_1-formula φ over $(A^*, \preccurlyeq, (w)_{w\in A^*})$ by [17, Theorem III.3]. Let $w_1, \ldots, w_\ell$ be the constants occurring in φ and consider the formula φ' in the Σ_1-fragment of $(A^*, \preccurlyeq)$ obtained by replacing each occurrence of w_j by a fresh variable y_j. Then $(u, w_1, \ldots, w_\ell)$ satisfies φ' if and only if $u \in L$. Thus, φ' defines an undecidable relation.

For the second statement, we claim that every language $L \subseteq A^*$ that is Σ_1-definable in $(A^*, \preccurlyeq)$ satisfies $A^*LA^* \subseteq L$. Hence, many automorphism-invariant regular languages such as $\bigcup_{a\in A} a^*$ are not definable. Note that for $a \in A$ and $u, v \in A^*$, we have $u \preccurlyeq v$ if and only if $au \preccurlyeq av$. Thus, every Σ_0-definable relation $R \subseteq (A^*)^k$ satisfies $(w_1, \ldots, w_k) \in R$ if and only if $(aw_1, \ldots, aw_k) \in R$. Symmetrically, $(w_1, \ldots, w_k) \in R$ is equivalent to $(w_1a, \ldots, w_ka) \in R$. As a projection of a Σ_0-definable relation, L thus satisfies $A^*LA^* \subseteq L$. ◀

5 Conclusion

We have shown how to define all recursively enumerable relations in the existential fragment of the subword order with constants for each alphabet A with $|A| \geq 3$. If $|A| = 1$, then the relations definable in $(A^*, \preccurlyeq, (w)_{w\in A^*})$ correspond to relations over $\mathbb{N}$ definable in $(\mathbb{N}, \leq)$ with constants. Hence, this case is very well understood: This structure admits quantifier elimination [30, Theorem 2.2(b)], which implies that the Σ_1-fragment is expressively complete and also that a subset of A^* is only definable if it is finite or co-finite. In particular, Theorem 2.1 does not hold for $|A| = 1$.

We leave open whether Theorem 2.1 still holds over a binary alphabet. If this is the case, then we expect that substantially new techniques are required. In order to express non-trivial relations over two letters, our proof often uses a third letter as a separator and marker for “synchronization points” in subword embeddings.

References

1 Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *information and computation*, 127(2):91–101, 1996.

2 Mohamed Faouzi Atig, Dmitry Chistikov, Piotr Hofman, K. Narayan Kumar, Prakash Saivasan, and Georg Zetzsche. The complexity of regular abstractions of one-counter languages. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 207–216. ACM, 2016. doi:10.1145/2933575.2934561.

3 Mohamed Faouzi Atig, Roland Meyer, Sebastian Muskalla, and Prakash Saivasan. On the upward/downward closures of Petri nets. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 49:1–49:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.MFCS.2017.49.

4 Ricardo A. Baeza-Yates. Searching subsequences. *Theoretical Computer Science*, 78(2):363–376, 1991.

5 Laura Barker, Pamela Fleischmann, Katharina Harwardt, Florin Manea, and Dirk Nowotka. Scattered factor-universality of words. In *Developments in Language Theory - 24th International Conference, DLT 2020, Tampa, FL, USA, May 11-15, 2020, Proceedings*, volume 12086 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 2020. doi:10.1007/978-3-030-48516-0_2.

6 David Barozzini, Lorenzo Clemente, Thomas Colcombet, and Pawel Parys. Cost automata, safe schemes, and downward closures. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 109:1–109:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.109.

7 Jean Berstel. *Transductions and Context-Free Languages*. Teubner, 1979.

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

9 Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1216–1235. SIAM, 2018. doi:10.1137/1.9781611975031.79.

10 Lorenzo Clemente, Pawel Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 96–105. ACM, 2016. doi:10.1145/2933575.2934527.

11 Joel D. Day, Pamela Fleischmann, Maria Kosche, Tore Koß, Florin Manea, and Stefan Siemer. The edit distance to k-subsequence universality. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 25:1–25:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.25.

12 Cees Elzinga, Sven Rahmann, and Hui Wang. Algorithms for subsequence combinatorics. *Theoretical Computer Science*, 409(3):394–404, 2008.

13 Lukas Fleischer and Manfred Kufleitner. Testing Simon's congruence. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 62:1–62:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.62.

14 Pawel Gawrychowski, Maria Kosche, Tore Koß, Florin Manea, and Stefan Siemer. Efficiently testing Simon's congruence. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 34:1–34:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.34.

15 Peter Habermehl, Roland Meyer, and Harro Wimmel. The downward-closure of petri net languages. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 466–477. Springer, 2010. doi:10.1007/978-3-642-14162-1_39.

16 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 151–163. ACM, 2016. doi:10.1145/2837614.2837627.

17 Simon Halfon, Philippe Schnoebelen, and Georg Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005141.

18 Simon Halfon, Philippe Schnoebelen, and Georg Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. *CoRR*, abs/1701.07470, 2017. arXiv:1701.07470.

19 Juris Hartmanis and John E Hopcroft. What makes some language theory problems undecidable. *Journal of Computer and System Sciences*, 4(4):368–376, 1970.

20 Prateek Karandikar, Manfred Kufleitner, and Philippe Schnoebelen. On the index of Simon's congruence for piecewise testability. *Inf. Process. Lett.*, 115(4):515–519, 2015. doi:10.1016/j.ipl.2014.11.008.

21 Prateek Karandikar and Philippe Schnoebelen. Decidability in the logic of subsequences and supersequences. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 84–97. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.84.

22 Prateek Karandikar and Philippe Schnoebelen. The height of piecewise-testable languages and the complexity of the logic of subwords. *Log. Methods Comput. Sci.*, 15(2), 2019. doi:10.23638/LMCS-15(2:6)2019.

23 Dexter C. Kozen. *Theory of computation.* Springer Verlag London Limited, 2010.

24 Oleg V. Kudinov, Victor L. Selivanov, and Lyudmila V. Yartseva. Definability in the subword order. In *Conference on Computability in Europe*, pages 246–255. Springer, 2010.

25 Dietrich Kuske. Theories of orders on the set of words. *RAIRO-Theoretical Informatics and Applications*, 40(01):53–74, 2006.

26 Dietrich Kuske and Christian Schwarz. Complexity of Counting First-Order Logic for the Subword Order. In Javier Esparza and Daniel Kráľ, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:12, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.61.

27 Dietrich Kuske and Georg Zetzsche. Languages ordered by the subword order. In *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 348–364. Springer, 2019. doi:10.1007/978-3-030-17127-8_20.

28 David Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM (JACM)*, 25(2):322–336, 1978.

29 Yuri Matiyasevich. *Hilbert's tenth problem.* MIT press, 1993.

30 Pierre Péladeau. Logically defined subsets of N^k. *Theoretical computer science*, 93(2):169–183, 1992.

31 Jacques Sakarovitch and Imre Simon. Subwords. In M. Lothaire, editor, *Combinatorics on Words*, Cambridge Mathematical Library, chapter 6, pages 105–142. Cambridge University Press, 2nd edition, 1997. doi:10.1017/CBO9780511566097.009.

32 Georg Zetzsche. An approach to computing downward closures. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2015. doi:10.1007/978-3-662-47666-6_35.

33 Georg Zetzsche. Computing downward closures for stacked counter automata. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 743–756. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.743.

34 Georg Zetzsche. The complexity of downward closure comparisons. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *Proc. of the 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 123:1–123:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Intrinsic Complexity of Recursive Functions on Natural Numbers with Standard Order

Nikolay Bazhenov
Sobolev Institute of Mathematics, Novosibirsk, Russia

Dariusz Kalociński[1]
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Michał Wrocławski
Faculty of Philosophy, University of Warsaw, Poland

Abstract

The intrinsic complexity of a relation on a given computable structure is captured by the notion of its degree spectrum – the set of Turing degrees of images of the relation in all computable isomorphic copies of that structure. We investigate the intrinsic complexity of unary total recursive functions on nonnegative integers with standard order. According to existing results, the possible spectra of such functions include three sets consisting of precisely: the computable degree, all c.e. degrees and all Δ_2 degrees. These results, however, fall far short of the full classification. In this paper, we obtain a more complete picture by giving a few criteria for a function to have intrinsic complexity equal to one of the three candidate sets of degrees. Our investigations are based on the notion of block functions and a broader class of quasi-block functions beyond which all functions of interest have intrinsic complexity equal to the c.e. degrees. We also answer the questions raised by Wright [21] and Harrison-Trainor [10] by showing that the division between computable, c.e. and Δ_2 degrees is insufficient in this context as there is a unary total recursive function whose spectrum contains all c.e. degrees but is strictly contained in the Δ_2 degrees.

2012 ACM Subject Classification Theory of computation → Computability

Keywords and phrases Computable Structure Theory, Degree Spectra, ω-Type Order, c.e. Degrees, d.c.e. Degrees, Δ_2 Degrees, Learnability

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.8

Related Version *Full Version*: `https://arxiv.org/abs/2110.01268`

Funding *Nikolay Bazhenov*: The work of Bazhenov is supported by the Mathematical Center in Akademgorodok under agreement No. 075-15-2019-1613 with the Ministry of Science and Higher Education of the Russian Federation.
Dariusz Kalociński: The work of Kalociński is supported by the National Science Centre Poland, grant No. 2018/31/B/HS1/04018.

Acknowledgements We would like to thank Matthew Harrison-Trainor, Aleksander Iwanow, Mars Yamaleev for helpful discussions and anonymous reviewers for comments.

1 Introduction

In mathematics we study structures of various sorts like rings, fields or linear orders. In computability theory we investigate the complexity of countable objects. A combination of the two – computable structure theory – examines the relationship between complexity and structure in the above sense [1, 14]. One of the main research programs in computable structure theory consists in the study of how complexity of a relation on a given structure

[1] `dariusz.kalocinski@gmail.com`

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 8; pp. 8:1–8:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

behaves under isomorphisms (see, e.g., [17, 9, 11, 7]). Recall that a structure is computable if its domain and basic relations are uniformly computable. The complexity of a relation can be captured by a measure such as Turing degrees. This leads to the notion of the degree spectrum (of a computable relation on a computable structure) – the set of Turing degrees assumed by the images of that relation in all computable isomorphic copies of that structure. This notion captures what might otherwise be called the intrinsic complexity of a relation.

A natural motivation for investigating intrinsic complexity comes from treating computable copies of a structure as notations: we regard the elements of the copy as names for the members of the structure, with the underlying isomorphism acting as a naming function. A computable copy of a structure is thus a notation in which all the basic relations are computable (meaning that their images within the copy are computable). This is essentially Shapiro's idea, as studied, though in a very restricted sense, in [18]. But this analogy goes further. Shapiro insisted, not without reason, that computations are not performed directly on numbers but rather on their names (using the terminology of computable structure theory: computations are not performed on the underlying structure but on isomorphic copies). This intuition transfers to all computation-dependent notions, including complexity. In the end, the intricate notion of intrinsic complexity boils down to the study of how difficult it is to compute the relation in notations in which all the basic relations are computable.

Following Downey et al. [5] and Wright [21], we investigate degree spectra on the most common ordering: non-negative integers with the standard *less than* relation, denoted by $(\omega, <)$. We study this question in the restricted setting of specific binary relations of general interest – graphs of unary total computable functions. As an example of how isomorphism might influence the complexity of a such a function, consider the successor. By a well-known result (see, e.g., Example 1.3 in [2]), there is an isomorphic copy of $(\omega, <)$ in which the image of the successor computes the halting problem. In general, however, as one can easily observe, the range of intrinsic complexity of a computable relation on $(\omega, <)$ is restricted to Δ_2 degrees and, therefore, each isomorphic image of such a relation is learnable (i.e., it possesses a recursive approximation) or, equivalently, Turing reducible to the halting problem [8, 16, 19].

Several results from the literature partially characterize degree spectra of such functions. Moses [15] provided a syntactical characterization of intrinsically computable (i.e. having only the computable degree in their spectrum) n-ary relations on $(\omega, <)$. These results imply that a total unary recursive function is intrinsically computable if and only if it is almost constant or almost identity (see Proposition 6). In [5], Downey, Khoussainov, Miller and Yu examined degree spectra of unary relations on $(\omega, <)$. Their results show, among others, that the spectrum of any infinite coinfinite computable unary relation on $(\omega, <)$ contains all c.e. degrees (Theorem 1.1 in [5]). Wright extended their results by showing the following.

▶ **Theorem 1** (Wright [21]). *The spectrum of a computable n-ary relation which is not intrinsically computable contains all c.e. degrees.*

He was also able to show that a computable *unary* relation which is not intrinsically computable has Δ_2 degrees as a spectrum (see, also, [12]).

Wright asked in [21] whether the computable, the c.e. and the Δ_2 degrees exhaust possible degree spectra for computable n-ary relations on $(\omega, <)$. Roughly at about the same time, Harrison-Trainor posed a related question in [10] where he showed that there exists a computable relation R on $(\omega, <)$ such that its degree spectrum either

(1) contains the c.e. degrees but does not contain all of the Δ_2 degrees, or

(2) consists of exactly all Δ_2 degrees but R does not have this degree spectrum uniformly.

Harrison-Trainor conjectured that (1) holds for the relation he constructed. We construct a unary total computable function (hence, a computable binary relation) witnessing (1). This also answers Wright's question.

Results of this paper are heavily based on certain structural characteristics of functions, which we refer to as the block and (a weaker) quasi-block property. Intuitively, each block function on $(\omega,<)$ is defined by multiple sub-functions where each sub-function applies to a different finite $<$-interval of ω (Definition 10). A quasi-block function is one for which there are increasingly long initial $<$-segments such that no number from within the segment is sent outside. The usefulness of these properties is clear in view of the observation that any computable non-quasi-block function has exactly all c.e. degrees as a spectrum (Theorem 18). One of the main contributions of the paper consists in the complete characterization of degree spectra of block functions which have at most finitely many isomorphism types of their elementary sub-functions (Theorem 14). The second main contribution is Theorem 23 which answers Wright's and Harrison-Trainor's questions.

2 Definitions

▶ **Definition 2.** *$(\omega,\prec)$ is a computable copy of $(\omega,<)$ if $\prec$ is a computable ordering on ω and structures $(\omega,<)$ and $(\omega,\prec)$ are isomorphic.*

▶ **Definition 3.** *Let R be a relation on $(\omega,<)$, i.e. $R\subseteq\omega^k$, for some $k\in\omega$, and let $\mathcal{A}$ be a computable copy of $(\omega,<)$. If φ is an isomorphism from $(\omega,<)$ to $\mathcal{A}$, we write $R_{\mathcal{A}}$ for the image of R under φ.*

▶ **Definition 4.** *Let R be a relation on $(\omega,<)$. The degree spectrum or spectrum of R on $(\omega,<)$, in symbols $DgSp_{(\omega,<)}(R)$, is the set of Turing degrees of $R_{\mathcal{A}}$ over all computable copies $\mathcal{A}$ of $(\omega,<)$.*

Throughout the article, we use abbreviated forms: *spectrum of R* and $DgSp(R)$.

▶ **Definition 5.** *Let R be a relation on $(\omega,<)$. The relation R is intrinsically computable if $DgSp(R)$ contains only the computable degree.*

Let $\mathcal{A}=(A,<_{\mathcal{A}})$ be a linear order. If $a\leq_{\mathcal{A}} b$, then $[a;b]_{\mathcal{A}}$ and $[a;b)_{\mathcal{A}}$ denote the intervals $\{x : a\leq_{\mathcal{A}} x\leq_{\mathcal{A}} b\}$ and $\{x : a\leq_{\mathcal{A}} x<_{\mathcal{A}} b\}$, respectively. If the order $\mathcal{A}$ is clear from the context, then we omit the subscript $\mathcal{A}$. *Succ* is the successor function on $(\omega,<)$. $\langle\cdot,\cdot\rangle$ is the pairing function. Computability-related notation is standard and follows [20]. For example, $\leq_T$ denotes the Turing reduction.

If $X\subseteq\omega$ is a Δ_2 set, then one can choose its *computable approximation* $\xi(k,s)$, i.e. a $\{0,1\}$-valued computable function such that $\lim_s\xi(k,s)=X(k)$, for all k. We often use notation $X_s(k)$ for $\xi(k,s)$.

3 Results

The following two statements will be useful (the proof of the first one is in the full version).

▶ **Proposition 6.** *Let f be a unary total computable function. Then f is intrinsically computable if and only if either f is almost constant, or f is almost identity.*

▶ **Proposition 7** (see, e.g., Example 1.3 in [2]). *The spectrum of successor is equal to the c.e. degrees.*

▶ **Theorem 8.** *Let f be a unary computable function with finite range. If f is not intrinsically computable then its spectrum is equal to the Δ_2 degrees.*

Proof. The proof is based on the ideas from Theorem 1.2 of [21]. We provide a detailed exposition, so that a reader could familiarize themselves with the proof techniques.

We fix $c_0 \neq c_1$ such that $f^{-1}(c_i)$ is infinite. Without loss of generality, one may assume that $c_0 = 0$ and $c_1 = 1$.

Let $X \subseteq \omega$ be an arbitrary Δ_2 set. We build a computable isomorphic copy $\mathcal{A} = (\omega, <_{\mathcal{A}})$ of the order $(\omega, <)$ such that $f_{\mathcal{A}}$ is Turing equivalent to the set X. Our construction will ensure that the following two conditions hold:

(i) $k \in X$ if and only if $f_{\mathcal{A}}(2k) = 1$, for all k;

(ii) the restriction of $f_{\mathcal{A}}$ to the set of odd numbers (i.e., $f_{\mathcal{A}} \upharpoonright \{2k+1 : k \in \omega\}$) is computable.

It is clear that these conditions imply $f_{\mathcal{A}} \equiv_T X$.

Let M be a large enough natural number such that

$$(\forall x > M)[\text{the } f\text{-preimage of } f(x) \text{ is infinite, and } x \notin \mathrm{range}(f)].$$

Beforehand, we use odd numbers to copy the initial segment $[0; M]$ of $(\omega, <)$. More formally, we put $2k + 1 <_{\mathcal{A}} 2l + 1$ for all $k < l \leq M$. In addition, any newly added (to the copy $\mathcal{A}$) number will be strictly $\mathcal{A}$-greater than $2M + 1$.

Our construction satisfies the following requirements:

$$\begin{aligned} e \in X \ &\Leftrightarrow\ f_{\mathcal{A}}(2e) = 1, \\ e \notin X \ &\Leftrightarrow\ f_{\mathcal{A}}(2e) = 0. \end{aligned} \qquad (\mathcal{R}_e)$$

As usual, this will be achieved by working with a computable approximation $X_s(e)$.

By $\mathcal{A}_s$ we denote the finite structure built at a stage s. At each stage s, there is a natural isomorphic embedding h_s from $\mathcal{A}_s$ into $(\omega, <)$. If $\mathcal{A}_s$ consists of $a_0 <_{\mathcal{A}} a_1 <_{\mathcal{A}} a_2 <_{\mathcal{A}} \ldots <_{\mathcal{A}} a_n$, then we assume that $h_s(a_i) = i$, for all $i \leq n$.

This convention allows one to talk about values $f_{\mathcal{A}_s}(x)$ for elements $x \in \mathcal{A}_s$. We simply assume that

$$f_{\mathcal{A}_s}(a_i) = h_s^{-1} \circ f \circ h_s(a_i).$$

Our construction will ensure that $f_{\mathcal{A}}(x) = \lim_s f_{\mathcal{A}_s}(x)$, for all x. Sometimes (when the usage context is unambiguous), we write just $f_{\mathcal{A}}(x)$ in place of $f_{\mathcal{A}_s}(x)$.

Strategy $\mathcal{R}_e$ in isolation. Suppose that $(s_0 + 1)$ is the first stage of work for this strategy. Then we add $2e$ to the right end of $\mathcal{A}$. Since we want to ensure that $f_{\mathcal{A}_{s_0+1}}(2e) = X_{s_0+1}(e)$, we also add (if needed) finitely many fresh odd numbers in-between $\mathcal{A}_s$ and $2e$, i.e., we set

$$a <_{\mathcal{A}} 2k + 1 <_{\mathcal{A}} 2e,$$

for $a \in \mathcal{A}_{s_0}$ and newly added numbers $2k + 1$.

We say that $\mathcal{R}_e$ *requires attention* at a stage s if the current value $f_{\mathcal{A}_s}(2e)$ is not equal to $X_s(e)$. In order to deal with $\mathcal{R}_e$, we introduce the following important ingredient of our proof techniques. For the sake of future convenience, we give a *general* description of the module.

Pushing-to-the-right module (PtR-module). We split the (current finite) structure $\mathcal{A}_s$ into three intervals: $B <_{\mathcal{A}} C <_{\mathcal{A}} D$, where, say, we have $B = [a; b]_{\mathcal{A}}$, $C = \{c^0 <_{\mathcal{A}} c^1 <_{\mathcal{A}} \ldots <_{\mathcal{A}} c^m\}$, and $D = \{d^0 <_{\mathcal{A}} d^1 <_{\mathcal{A}} \ldots <_{\mathcal{A}} d^n\}$. Informally speaking, the module aims to achieve the following goal: while preserving all values $f_{\mathcal{A}}(x)$ for $x \in B \cup D$, we want to change the function $f_{\mathcal{A}} \upharpoonright C$ in such a way that $f_{\mathcal{A}}$ satisfies a particular requirement. In addition, we require that C remains an interval inside $\mathcal{F}$.

More formally, we extend the structure $\mathcal{A}_s$ to a finite structure $\mathcal{F}$ (which is intended to be an initial segment of $\mathcal{A}_{s+1}$) with the following properties:

- every element $x \in \mathcal{F} \setminus \mathcal{A}_s$ is a fresh odd number, and each such x satisfies either $B <_{\mathcal{A}} x <_{\mathcal{A}} C$ or $x >_{\mathcal{A}} C$;
- $f_{\mathcal{F}}(d^i) = f_{\mathcal{A}_s}(d^i)$ for all $i \leq n$;
- the new values $f_{\mathcal{F}}(c^j)$ satisfy some *target condition*.

In the future, when we talk about a particular instance of the module, we will always explicitly specify the desired target condition.

Roughly speaking, our module keeps the interval B fixed, while all elements from $C \cup D$ are pushed to the right (with the help of newly added odd numbers). In addition, the elements of C stick together.

Going back to $\mathcal{R}_e$: if $\mathcal{R}_e$ requires attention at a stage s, then we implement the following actions.

The PtR-module for the strategy $\mathcal{R}_e$. In our $\mathcal{R}_e$-setting, we choose the middle interval C as the singleton $\{2e\}$. The desired target condition is a natural one: we aim to satisfy $f_{\mathcal{A}}(2e) = X_s(e)$.

We build a finite structure $\mathcal{F}$ extending $\mathcal{A}_s$ as dictated by the PtR-module. Then we declare that $\mathcal{F}$ is the output of our module, and proceed further. This concludes the description of the $\mathcal{R}_e$-strategy.

Construction. At a stage $s+1$, we work with strategies $\mathcal{R}_e$, for $e \leq s$. So, a strategy $\mathcal{R}_e$ starts working at the stage $e+1$. For each $\mathcal{R}_e$ (in turn), our actions follow the description given above. After $\mathcal{R}_i$ finished its work, the PtR-module of the next strategy $\mathcal{R}_{i+1}$ works with the finite structure produced by $\mathcal{R}_i$. Since the described PtR-module preserves $f_{\mathcal{A}} \upharpoonright (B \cup D)$, our strategies do not injure each other. We define $\mathcal{A} = \bigcup_{s \in \omega} \mathcal{A}_s$, where $\mathcal{A}_{s+1}$ is the final content of our structure produced by the PtR-module of $\mathcal{R}_s$ at the end of stage $s+1$.

Verification. First, we show that in the construction, every application of a PtR-module is successful (i.e., one can always build a desired structure $\mathcal{F}$).

In order to prove this, we consider our structures from a different angle: the structure $(\omega, <, f)$ can be treated as an infinite string β over a finite alphabet $\Sigma = \text{range}(f)$, where the i-th symbol $\beta(i)$ of the string is equal to $f(i)$, $i \in \omega$.

Then the construction of $\mathcal{F}$ in the PtR-module can be re-interpreted as follows. We are given three finite strings, namely σ, τ (of length one), and ρ (of length $n+1$), for the intervals B, $C = \{2e\}$, and D correspondingly. Our task is to find finite strings $\tau', \rho'_0, \rho'_1, \ldots, \rho'_n$ with the following property:

$$\sigma\, \tau'\, a\, \rho'_0\, \rho(0)\, \rho'_1\, \rho(1)\, \ldots\, \rho'_n\, \rho(n),$$

where $a = X_s(e)$, is an initial segment of β.

This task can be always implemented successfully – this is a consequence of the following simple combinatorial fact.

▶ Remark 9. Let Σ be a finite alphabet, and let $\alpha \in \Sigma^{\omega}$ be an infinite string over Σ. Suppose that every symbol from Σ occurs infinitely often in α. Then for every finite string $\sigma \in \Sigma^{<\omega}$ of length $m > 0$, one can find finite strings $\tau_0, \tau_1, \dots, \tau_{m-1}$ such that

$\tau_0\, \sigma(0)\, \tau_1\, \sigma(1)\, \dots\, \tau_{m-1}\, \sigma(m-1)$ is an initial segment of α. ⌟

So, we deduce that all applications of a PtR-module are successful. Hence, if $e \leq s$, then by the end of the stage $s+1$ we have $f_{\mathcal{A}_{s+1}}(2e) = X_s(e)$. This implies that every requirement $\mathcal{R}_e$ is satisfied.

Each element $a \in \mathcal{A}$ moves (to the right) only finitely often. Indeed, there are only finitely many even numbers $2e$ such that $2e \leq_{\mathcal{A}} a$. Consider a stage s^* such that the values $X_s(e)$ (for these $2e$) never change after s^*. Clearly, the element a never moves after the stage s^*.

We deduce that the structure $\mathcal{A}$ is a computable copy of $(\omega, <)$. For every k, after the value $f_{\mathcal{A}_s}(2k+1)$ is defined for the first time, this value never changes (since the PtR-module always preserves the restriction $f_{\mathcal{A}} \upharpoonright (B \cup D)$). Therefore, our structure $\mathcal{A}$ satisfies Conditions (i) and (ii) defined above. Theorem 8 is proved. ◀

3.1 Block and Quasi-Block Functions

From now on, we study some natural subclasses of unary total recursive functions with infinite range.

▶ **Definition 10.** *Let $f\colon \omega \to \omega$ be a total function. An interval I of $(\omega, <)$ is* f-closed *if for all $x \in I$, $f(x) \in I$ and $f^{-1}(x) \subseteq I$. For a finite non-empty interval $I \subset \omega$, the structure $(I, <, f \upharpoonright I)$ is an* f-block *if it has the following properties:*

- *I is an f-closed interval and it cannot be written as a disjoint union of several f-closed intervals;*
- *$\{x \in \omega : x < I\}$ is f-closed.*

The function f is a block function *if for every $a \in \omega$, there is an f-block containing a.*
If $(I, <, f \upharpoonright I)$ is an f-block, we refer to its isomorphism type as an f-type *(or a type).*

The second condition of the definition above ensures that for a block function f, every element is contained in a unique f-block. Observe that in Fig. 1, without this condition, the element 2 would be an f-block itself, which we would like to avoid.

▶ Remark 11. For any computable block function f there is a 1-1 computable enumeration of its types. f can be represented by the unique infinite string $\alpha_f : \omega \to [0; N)$, where $[0, N)$ is the domain of the enumeration, for some $N \in \omega \cup \{+\infty\}$. For example, if $I_0, I_1, \dots, I_N$ are all (isomorphism types of) f-blocks, then $(\omega, <, f)$ can be treated as an infinite string $\alpha_f : \omega \to \{n : 0 \leq n \leq N\}$, e.g. a string 012012012... corresponds to a disjoint sum of the following form: $I_0 + I_1 + I_2 + I_0 + I_1 + I_2 + I_0 + I_1 + I_2 + \dots$

▶ **Example 12.** $f(n) = 2 \cdot \lfloor \frac{n}{2} \rfloor$ is a block function. Its spectrum consists of all Δ_2 degrees by Theorem 14 below.

▶ **Example 13.** Consider finite structures $\mathcal{J}_n$ from Figure 1. Let g be the involution such that $(\omega, <, g) \cong \mathcal{J}_0 + \mathcal{J}_1 + \mathcal{J}_2 + \dots$ Clearly, g is a block function. In the full version, we show that its degree spectrum is all of the c.e. degrees.

▶ **Theorem 14.** *Let f be a computable block function such that it has only finitely many f-types and f is not almost identity. Then the spectrum $DgSp(f)$ consists of all Δ_2 degrees.*

Due to space constraints, the proof has been moved to the appendix.

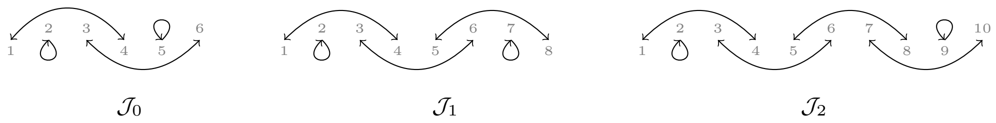


Figure 1 Structures $\mathcal{J}_n = ([1; 6+2n], <, f)$, for $n = 0, 1, 2$, where f is the involution such that $f(k) = k$ iff $k = 2$ or $k = 6+2n-1$, and $f(k) = k+3$ for odd numbers $\leq 6+2n-3$.

The notion of a quasi-block function is a generalization of the notion of a block function. Unlike blocks which are disjoint and follow each other, quasi-blocks are increasingly larger and they are initial segments of ω.

▶ **Definition 15.** *We say that $f : \omega \to \omega$ is a* quasi-block function *if there are arbitrarily long finite initial segments of ω closed under f. For any such segment $I = [0; n]$, the structure $(I, <, f \restriction I)$ is an f-*quasi-block*.*

If f is a quasi-block function but not a block function, we call f a proper quasi-block function.

▶ **Example 16.** Euler's function is a function φ such that if $n > 0$, then $\varphi(n)$ is the number of such $m \leq n$ that m and n are relatively prime. φ is a proper quasi-block function. Since φ has a computable non-decreasing lower bound $\lfloor \sqrt{\frac{n}{2}} \rfloor$ diverging to ∞ (see, e.g., [13, p. 9]), the spectrum of φ is equal to the c.e. degrees by Theorem 19.

▶ **Example 17.** The function $nd : \omega \to \omega$ assigning to each $n > 0$ the number of its divisors is a proper quasi-block function.

Below we describe a method used to show that the degree spectrum of a certain unary recursive function f consists exactly of c.e. degrees.

Retrieving the Successor module (RS) on $(\omega, <, f)$, for f recursive, is a scheme of algorithms which, for any computable copy $\mathcal{A}$ of $(\omega, <)$ and an initial segment $\mathcal{I}_t$ of $\mathcal{A}$ satisfying some condition $\mathcal{R}$ (to be specified in a concrete implementation) computes, uniformly in t and relative to $f_{\mathcal{A}}$, a longer initial segment $\mathcal{I}_{t+1}$ of $\mathcal{A}$ satisfying $\mathcal{R}$, which enables us to construct an increasing sequence of initial segments $\mathcal{I}_0 \subset \mathcal{I}_1 \subset \dots$.

Suppose that there exists a concrete implementation of the RS-module for $(\omega, <, f)$. We wish to show that the degree spectrum of f on $(\omega, <)$ consists of exactly c.e. degrees. To this aim, we want to show that $Succ_{\mathcal{A}}$ is Turing-reducible to $f_{\mathcal{A}}$. We also observe that the reduction in the other direction works. We conclude that $Succ_{\mathcal{A}} \equiv_T f_{\mathcal{A}}$, hence $DgSp(Succ) = DgSp(f)$, i.e. they consist of all c.e. degrees. This conclusion is based on Proposition 7.

Suppose that an initial segment of ω up to n (according to $<$) has already been determined, along with its isomorphic image $\mathcal{I}_t$ in $(\omega, \prec)$. Let us adopt a convention that the isomorphic image of each number i is k_i. Observe that for each number i such that $k_i \prec k_n$ we know how to determine its successor in $(\omega, \prec)$. In an application of the RS-module, given k_n – the rightmost element of $\mathcal{I}_t$ – we get some k_m and m such that $k_n \prec k_m$ and $[k_0; k_m]_{\mathcal{A}}$ satisfies $\mathcal{R}$. We know that in $\mathcal{A}$ there are exactly $m - n - 1$ elements between k_n and k_m. Since the ordering $\prec$ is recursive, we can check elements one by one until we determine what elements (and in what order) are between k_n and k_m. This way we extend the initial segment $\mathcal{I}_t$ of $\mathcal{A}$ to a larger initial segment $\mathcal{I}_{t+1}$ satisfying $\mathcal{R}$ and we are able to retrieve more values of the successor in this structure.

▶ **Theorem 18.** *The spectrum of any unary total computable non-quasi-block function is equal to the c.e. degrees.*

Proof. We show that the RS module can be used for $(\omega, <, f)$. Given a computable copy $\mathcal{A}$ of $(\omega, <)$, we set $\mathcal{I}_0$ as the image of some initial segment of $(\omega, <)$ such that for every position n outside of $\mathcal{I}_0$ there is $m \leq n$ such that $f(m) > n$. The condition $\mathcal{R}$ states that there is a position j within $\mathcal{I}_t$ such that $f(j) > n$. Then if we already know $\mathcal{I}_t$ and want to determine $\mathcal{I}_{t+1}$, we calculate both $f(j)$ and $f_{\mathcal{A}}(k_j)$ from the condition $\mathcal{R}$, obtaining some values of these functions m and k_m, each of them somewhere behind n and k_n in their sequences. ◀

▶ **Theorem 19.** *If f is a recursive proper quasi-block function with a computable non-decreasing lower bound diverging to $+\infty$, then its spectrum consists of exactly c.e. degrees.*

Proof. We claim that there exist only finitely many quasi-blocks closed under both f and f^{-1}. Observe that if there were infinitely many such quasi-blocks, then f would be a block function. Observe also that if f is as above, then we are able to calculate how many times each of its values is assumed.

We utilise the RS module. The segment $\mathcal{I}_0$ is any initial segment such that none of its super-quasi-blocks is closed under both f and f^{-1}. Assume we already have a segment $\mathcal{I}_t$ of $\mathcal{A}$ retrieved. $\mathcal{I}_t$ satisfies the condition $\mathcal{R}$ stating that it is an initial segment which is not closed under both f and f^{-1}.

We wish to algorithmically construct $\mathcal{I}_{t+1}$, a segment of $\mathcal{A}$, satisfying the same condition $\mathcal{R}$. If there is $n \in \mathcal{I}_t$ such that $f_{\mathcal{A}}(n) >_{\mathcal{A}} \mathcal{I}_t$, we set $\mathcal{I}_{t+1}$ as the segment consisting of all elements up to $f_{\mathcal{A}}(n)$. If not, then there must be $m \in \mathcal{I}_t$ such that for some $n >_{\mathcal{A}} \mathcal{I}_t$, $f_{\mathcal{A}}(n) = m$. What is more, for every such m there are only finitely many arguments satisfying this identity and we are able to determine what they are (by looking at their isomorphic images in the standard copy). If M is the largest of these elements, then we set $\mathcal{I}_{t+1}$ as the segment until M. ◀

▶ **Theorem 20.** *There exists a recursive proper quasi-block function f with a non-decreasing lower bound diverging to $+\infty$ but with no such computable bound, with all c.e. degrees as a spectrum.*

Proof. Consider a set $A \subseteq \omega$ which is Δ_2 but not computable. Observe that for each such set there is a recursive sequence g of natural numbers such that each natural number appears in g at most finitely many times and for any $n \in \omega$, $n \in A$ iff the number of occurrences of n in g is odd.

f is going to be g modified in such a way that we put some fixed points between elements of g, pushing these elements to the right, to ensure that f is a quasi-block function. We will be able to easily distinguish (within f) old elements of g from the new filler elements, because only the new elements are going to be fixed points of f.

We construct f by finite extension, starting from the empty function. Initially, all elements of sequence g are unused. At any given stage, suppose that $g(m)$ is the least unused element of sequence g and that n is the least argument such that $f(n)$ is not defined yet. If $g(m) \geq n$, then for each $i = n, \ldots, g(m)$ assign $f(i) = i$. Regardless of whether you performed the previous instruction, assign value $g(m)$ to the least i such that $f(i)$ has no value set yet. We declare that $g(m)$ is used and go to the next stage.

This is a quasi-block function because each argument n is either a fixed point or is a number from sequence g which has been pushed so far to the right that $f(n) < n$. Hence every finite initial segment of ω is closed under f. However, this is not a block function. If it were, then every m such that $f(m) = n$ would need to be in the same block as n. Then we would be able to count how many times n is assumed as the value of f and hence A would be decidable.

The lower bound of f diverges to ∞ because every value can be assumed only finitely often. However, no such bound is computable because otherwise we would be able to determine the last occurrence of every number in g and A would be computable. Observe we can assume that this bound is non-decreasing. We just need to set $f(n) =$ the largest m such that $f(i) \geq m$ whenever $i \geq n$.

If A is a c.e. set, then we utilise the RS module to show that the degree spectrum of f consists of exactly the c.e. degrees. We can assume without loss of generality that g assumes each of its values only once, then so does f if we ignore fixed points.

We take $\mathcal{I}_0$ such that behind it there are no quasi-blocks closed under f^{-1}. The condition $\mathcal{R}$ states that there is an element $n > \mathcal{I}_t$ such that $f(n) \in \mathcal{I}_t$. Observe that such element is determined uniquely. We want to retrieve $\mathcal{I}_{t+1} \supseteq \mathcal{I}_t$ satisfying $\mathcal{R}$. We need to look for n described above and then to fill in all the missing numbers between $\mathcal{I}_t$ and n. Since the segment thus obtained is not a block, it needs to satisfy $\mathcal{R}$. We call this segment $\mathcal{I}_{t+1}$. ◀

3.2 Unusual Degree Spectrum

In this section we answer Wright's question (Question 6.2 in [21]). The result we prove here is also relevant for Harrison-Trainor's question (p. 5 in [10]). Recall a representation of a block function f as an infinite sequence α_f of (the indices of) types (see, Remark 11).

▶ **Definition 21.** *Let f be a computable block function with infinitely many types. The counting function for f is defined by $c_f(n) = \#\{i : \alpha_f(i) = n\}$.*

▶ **Proposition 22.** *Let f be a computable block function with all types pairwise non-embeddable, each occurring finitely often. Then* $\deg(c_f)$ *is c.e. and $f_{\mathcal{A}} \geq_T c_f$ implies that $deg(f_{\mathcal{A}})$ is c.e.*

Proof. $C_f^{\leq} := \{(k, n) : k \leq c_f(n)\}$ is c.e., $C_f^{\geq} := \{(k, n) : k \geq c_f(n)\}$ is co-c.e., so $\deg(C_f^{\leq} \oplus C_f^{\geq})$ is c.e. Since $C_f^{\leq} \oplus C_f^{\geq} \equiv_T c_f$, c_f is of c.e. degree.

Assume that $f_{\mathcal{A}} \geq_T c_f$. Then this implies $Succ_{\mathcal{A}} \leq_T f_{\mathcal{A}}$. Indeed, this fact can be illustrated by an example: with the oracle $f_{\mathcal{A}}$, one could recover that the structure $\mathcal{A}$ has, say, precisely two cycles of size 7. Since such a cycle is not embeddable into any other f-block, we could compute the precise positions of the two $f_{\mathcal{A}}$-cycles of size 7 (by looking at the standard copy of $\mathcal{A}$). Suppose that b is the rightmost element of the rightmost $f_{\mathcal{A}}$-cycle of size 7. Using this information, we could recover the values $Succ_{\mathcal{A}}(x)$ for all $x <_{\mathcal{A}} b$. Since all f-types are pairwise non-embeddable, we can "iterate" this process and compute $Succ_{\mathcal{A}}(x)$ for all x.

Note that $Succ_{\mathcal{A}} \geq_T f_{\mathcal{A}}$ always (for a computable f). Hence $f_{\mathcal{A}} \equiv_T Succ_{\mathcal{A}}$, and thus, by Proposition 7, $f_{\mathcal{A}}$ is of c.e. degree. ◀

▶ **Theorem 23.** *There exists a total computable function whose degree spectrum strictly contains all c.e. degrees and is strictly contained in the Δ_2 degrees.*

We construct a computable block function f with infinitely many types and each $c_f(n)$ finite. We want $c_f <_T \mathbf{0}'$ and a computable copy $\mathcal{A}$ of $(\omega, <)$ with $f_{\mathcal{A}}$ of non-c.e. degree. Combining this with Proposition 22 and a result by Cooper, Lempp and Watson from [4] (see Theorem 29) finishes the proof.

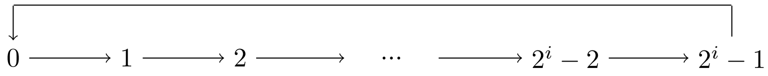


Figure 2 $\mathcal{C}_i = ([0; 2^i - 1], <, f_i)$, where the order $<$ is standard and f_i corresponds to the arrows.

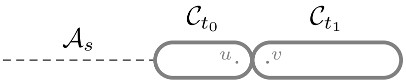


Figure 3 $\mathcal{A}_{s+1}$ after reserving $\langle u, v\rangle$ and tickets t_0, t_1, t_2 for $\mathcal{R}_{\langle e_1, e_2, n\rangle}$.

For each $e, e_1, e_2, n \in \omega$, we have the following requirements:

$$\mathcal{I}_e : I \not\leq \Phi_e^J, \qquad \mathcal{J}_e : J \not\leq \Phi_e^I, \qquad \text{and} \qquad R_{\langle e_1, e_2, n\rangle} : \Phi_{e_1}^{\Gamma_{f_\mathcal{A}}} \not\simeq W_n \vee \Phi_{e_2}^{W_n} \not\simeq \Gamma_{f_\mathcal{A}},$$

where $\Gamma_{f_\mathcal{A}}$ is the graph of $f_\mathcal{A}$. J is to make I incomplete while I is going to compute c_f. The non-c.e. degree requirements are based on [6, p. 195] (see, also, [3]).

At stage s we have finite sets I_s, J_s, and a finite structure $\mathcal{A}_s = (A_s, <_{\mathcal{A}_s}, f_{\mathcal{A}_s})$ with $f_{\mathcal{A}_s} : A_s \to A_s$ total. Eventually, we set $\mathcal{A} = \bigcup_{s \in \omega} \mathcal{A}_s$. We assume some recursive ω-type ordering of $\mathcal{I}_e$, $\mathcal{J}_e$, $\mathcal{R}_{\langle e_1, e_2, n\rangle}$, for all $e, e_1, e_2, n \in \omega$. During construction, requirements reserve numbers and, in order to be satisfied, they wait until those numbers meet certain conditions, in which case we say that they need attention.

- $\mathcal{I}_e$ (or $\mathcal{J}_e$) needs attention at stage $s+1$, if some x reserved for it at stage s and $I_s(x) = \Phi_{e,s}^{J_s}$ (or $J_s(x) = \Phi_{e,s}^{I_s}$).
- $R_{\langle e_1, e_2, n\rangle}$ needs attention at stage $s+1$ if, at stage s, some $\langle u, v\rangle$ is reserved for it, along with certain t_0, t_1, t_2 (called tickets), and, for some z, $\langle u, v\rangle < z < s$:

$$(\alpha)\ \Phi_{e_1,s}^{\Gamma_{f_{\mathcal{A}_s}}}[z] = W_{n,s}[z] \qquad \text{and} \qquad (\beta)\ \Phi_{e_2,s}^{W_{n,s}[z]}(\langle u, v\rangle) = \Gamma_{f_{\mathcal{A}_s}}(\langle u, v\rangle).$$

We use a variant of PtR (the proof of Theorem 8). In each application of PtR we distinguish E – the set of fresh numbers – for which we formulate an additional E-condition.

3.2.1 Construction

Let $(\mathcal{C}_i)_{i\in\omega}$ be a computable sequence of cycles, where $\mathcal{C}_i$ is of length 2^i (Figure 2). Put $I_0 = J_0 = \emptyset, \mathcal{A}_0 = (\emptyset, \emptyset, \emptyset)$. Requirements have no reserved numbers, no numbers are frozen. Below we describe stage $s+1$, for $s \in \omega$.

1. If no requirement needs attention at stage $s+1$, we choose the highest priority requirement with no reservation. If this is some $\mathcal{I}_e$ (or $\mathcal{J}_e$), we reserve for it the least fresh number x. If the highest priority requirement with no reservation is some $\mathcal{R}_{\langle e_1, e_2, n\rangle}$, we reserve for it the least number $\langle u, v\rangle$, fresh for $\mathcal{A}_s$ (i.e. u, v do not occur in $\mathcal{A}_s$), and three *consecutive* fresh numbers t_0, t_1, t_2, called tickets. We apply PtR by setting $B = \mathcal{A}_s$, $C = D = \emptyset$ and $E \supseteq \{u, v\}$ such that $|E| = 2^{t_0} + 2^{t_1}$ with every $x \in E$ being fresh for $\mathcal{A}_s$. We build a structure $\mathcal{E} = (E, <_\mathcal{E}, g)$ where $<_\mathcal{E}$ is a linear order satisfying the E-condition, depicted in Figure 3, which is:
   - $C_{t_0} + \mathcal{C}_{t_1} \cong \mathcal{E}$,
   - u is the $<_\mathcal{E}$-last element in the block corresponding to $\mathcal{C}_{t_0}$, and
   - v the $<_\mathcal{E}$-first element in the block corresponding to $\mathcal{C}_{t_1}$.

   We set $\mathcal{A}_{s+1} = \mathcal{A}_s + \mathcal{E}$. We have $\langle u, v\rangle \notin \Gamma_{f_{\mathcal{A}_{s+1}}}$. We enumerate ticket t_0 into I.

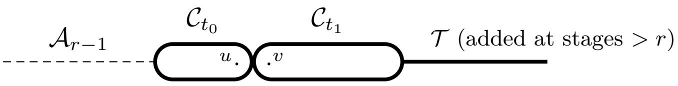


Figure 4 $\mathcal{A}_s$ when $R_{\langle e_1,e_2,n\rangle}$ receives attention for the first time with $\langle u,v\rangle$ and tickets t_0,t_1,t_2, assuming that the reservation has been made at stage r.

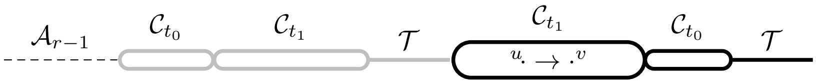


Figure 5 The result of reaction to *first attention* for $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ with reservation $\langle u,v\rangle$ and tickets t_0,t_1,t_2. Gray part is occupied by fresh numbers, thick part represents pushed numbers.

2. If a requirement needs attention, pick the highest one. We say it receives attention. If this is $\mathcal{I}_e$, some x is reserved for $\mathcal{I}_e$ at stage s and $I_s(x) = \Phi^{J_s}_{e,s}(x)$. Put x into I, freeze the computation $\Phi^{J_s}_{e,s}(x)$ and cancel all freezings and reservations for lower priority requirements. Deal with with $\mathcal{J}_e$ accordingly.
Suppose the highest priority requirement needing attention is some $\mathcal{R}_{\langle e_1,e_2,n\rangle}$. Some $\langle u,v\rangle$ is reserved for $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ at stage s with some tickets t_0,t_1,t_2. Below we describe reactions to first and second attention received by $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ with reservation $\langle u,v\rangle, t_0,t_1,t_2$.

(i) Suppose the reservation for $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ has been made at stage r. After r and before $s+1$ the structure $\mathcal{A}$ might have been extended by some $\mathcal{T}$ (thick line in Figure 4). The idea is that we push to the right all numbers that occupy the highlighted positions in Figure 4 and obtain the structure as in Figure 5.
More formally, divide $\mathcal{A}_s$ into $\mathcal{A}_s = \mathcal{B}+\mathcal{C}+\mathcal{D}$, where $\mathcal{B} \cong \mathcal{A}_{r-1}$, $\mathcal{C} \cong \mathcal{C}_{t_0}+\mathcal{C}_{t_1}$ and $\mathcal{D} \cong \mathcal{T}$, and apply PtR. Take $|C \cup D|$ numbers, fresh for $\mathcal{A}_s$, and make F out of them. Build a structure $\mathcal{F} = (F, <_{\mathcal{F}}; g)$, where $<_{\mathcal{F}}$ is a linear order, satisfying the F-condition $\mathcal{F} \cong \mathcal{C}+\mathcal{D}$. We rebuild $\mathcal{C}$ to get $\mathcal{C}' = (C, <_{\mathcal{C}}; h)$ where $\mathcal{C}'$ satisfies the C-condition $\mathcal{C}' \cong \mathcal{C}_{t_1}+\mathcal{C}_{t_0}$. We set $\mathcal{A}_{s+1} = \mathcal{B}+\mathcal{F}+\mathcal{C}'+\mathcal{D}$ (Figure 5).
Observe that pushed numbers from $\mathcal{C}+\mathcal{D}$ assume in $\mathcal{A}_{s+1}$ the same structure as in $\mathcal{A}_s$ except that the behavior of $f_{\mathcal{A}_{s+1}}$ (on numbers from $\mathcal{C}$) mimics $\mathcal{C}_{t_1}+\mathcal{C}_{t_0}$. This makes $\Gamma_{f_{\mathcal{A}_{s+1}}}(\langle u,v\rangle) = 1$ and thus $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ is satisfied at stage $s+1$. We enumerate t_1 into I and invalidate all reservations and freezings for lower priority requirements.

(ii) Suppose $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ has made the reservation at stage r and received the first attention at stage $p+1$. By the time we got to stage $s+1$, the structure $\mathcal{A}$ might have been extended by some $\mathcal{U}$ (Figure 6).
The idea is that we push all numbers occupying the highlighted positions in Figure 6 and obtain the structure as in Figure 7.
More formally, we divide $\mathcal{A}_s = \mathcal{B}+\mathcal{C}+\mathcal{D}$ in a way that $\mathcal{B} \cong \mathcal{A}_{r-1}+\mathcal{C}_{t_0}+\mathcal{C}_{t_1}+\mathcal{T}$, $\mathcal{C} \cong \mathcal{C}_{t_1}+\mathcal{C}_{t_0}$ and $\mathcal{D} \cong \mathcal{T}+\mathcal{U}$ with u,v residing in a copy of $\mathcal{C}_{t_1}$ within C. We apply PtR with $\mathcal{B},\mathcal{C},\mathcal{D}$ defined above. Let F be the set of $|C \cup D|$ numbers,

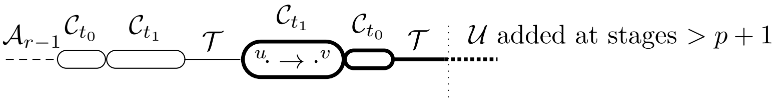


Figure 6 $\mathcal{A}_s$ when $R_{\langle e_1,e_2,n\rangle}$ receives attention for the second time with $\langle u,v\rangle$ and tickets t_0,t_1,t_2, assuming that the reservation has been made at stage r.

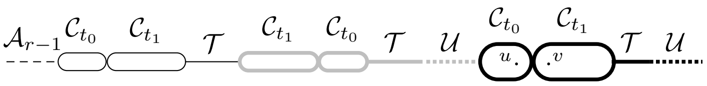


Figure 7 The result of reaction to *second attention* of $\mathcal{R}_{\langle e_1,e_2,n\rangle}$. Gray part is occupied by fresh numbers, thick part represents pushed numbers.

fresh for $\mathcal{A}_s$. We build a finite structure $\mathcal{F} = (F, <_{\mathcal{F}}, g)$, where $<_{\mathcal{F}}$ is a linear order, satisfying the F-condition $\mathcal{F} \cong \mathcal{C} + \mathcal{D}$. We rebuild $\mathcal{C}$ to get $\mathcal{C}' = (C, <_{\mathcal{C}}, h)$ satisfying the C-condition $\mathcal{C}' \cong C_{t_0} + C_{t_1}$. We set $\mathcal{A}_{s+1} = \mathcal{B} + \mathcal{F} + \mathcal{C}' + \mathcal{D}$. We have $\Gamma_{f_{\mathcal{A}_{s+1}}}(\langle u, v\rangle) = 0$. $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ is satisfied at stage $s+1$. We enumerate t_2 into I and invalidate all reservations and freezings for lower priority requirements.

3.2.2 Verification

Due to space constraints, the proof of the following lemma can be found in the full version.

▶ **Lemma 24.** *$\mathcal{A}$ is computable.*

▶ **Lemma 25.** *Every requirement is eventually satisfied. Hence, I, J are intermediate and $f_{\mathcal{A}}$ is of non-c.e. degree.*

Proof. This follows from finite injury. It remains to observe that each requirement can receive attention only finitely many times with the same numbers reserved for it. This is clear for $\mathcal{I}_e, \mathcal{J}_e$ (see, e.g. [20, Chap. VII.2]). We show that no $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ needs attention more than twice with the same $\langle u, v\rangle$ and tickets t_0, t_1, t_2 reserved for it. The PtR modules that we use to satisfy each $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ are carefully arranged to make the standard pattern of verification work (cf. [6, p. 196]). Suppose the reservation was made at stage r, the first attention was at stage $s+1$ and the second at stage $t+1$. Since $\langle u, v\rangle, t_0, t_1, t_2$ are reserved for $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ at stage $t \geq s+1$, no requirement with lower priority than $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ has received attention at any stage u, $t \geq u \geq s+1$. Actions performed at stage $t+1$ lead to $\mathcal{A}_{t+1} \restriction A_s = \mathcal{A}_s \restriction A_s$ (where A_s is the domain of $\mathcal{A}_s$). Therefore, $\Phi_{e_1}^{\Gamma_{f_{\mathcal{A}_{t+1}\restriction A_s}}}[z] = \Phi_{e_1}^{\Gamma_{f_{\mathcal{A}_s\restriction A_s}}}[z] = \Phi_{e_1}^{\Gamma_{f_{\mathcal{A}_s}}}[z] = W_{n,s}[z]$. At stage $s+1$ we had $\Phi_{e_2}^{W_{n,s}[z]}(\langle u, v\rangle) = \Gamma_{f_{\mathcal{A}_s}}(\langle u, v\rangle) \neq \Gamma_{f_{\mathcal{A}_t}}(\langle u, v\rangle)$. Since at stage $t+1$ we had $\Phi_{e_2}^{W_{n,t}[z]}(\langle u, v\rangle) = \Gamma_{f_{\mathcal{A}_t}}(\langle u, v\rangle)$ we must have $W_{n,t}[z] \neq W_{n,s}[z]$. Hence, for some x, $\Phi_{e_1}^{\Gamma_{f_{\mathcal{A}_{t+1}\restriction A_s}}}(x) = W_{n,s}(x) \neq W_{n,t}(x)$. Now, observe that $\mathcal{A}_{t+1} \restriction A_s$ does not change at any later stage at which $\langle u, v\rangle$ is reserved for $\mathcal{R}_{\langle e_1,e_2,n\rangle}$. Hence, for all such stages $w \geq t+1$, $\Phi_{e_1,w}^{\Gamma_{f_{\mathcal{A}_w\restriction A_s}}}(w) \neq W_{n,w}(x)$ and $\mathcal{R}_{\langle e_1,e_2,n\rangle}$ does not need attention at stage $w+1$. ◀

▶ **Lemma 26.** *For every $n \in \omega$, $c_f(n)$ is finite and is never increased due to numbers $> n+2$ entering I.*

Proof. Suppose the contrary. Then there exists n such that $c_f(n)$ is increased because of some $k > n+2$ entering I. Let $s+1$ be the stage at which this happens. Since $c_f(n)$ is increased at stage $s+1$, $\mathcal{C}_n$ is present in $\mathcal{A}_{s+1}$. Since $c_f(n)$ is increased due to k entering I, k must be associated at stage $s+1$ with some R_i. Hence, k is one of the tickets t_0, t_1, t_2 paired with R_i at this point. There are three cases.

($k = t_0$) This is when R_i is initialized and receives tickets t_0, t_1, t_2 (see Figure 3). For $c_f(n)$ to increase, we must have $n = t_0$ or $n = t_1$. $n = t_0$ is not possible because then we would have $k = t_0 = n$ which contradicts $k > n+2$. $n = t_1$ is also not possible because we would have $k = t_0 = t_1 - 1 = n - 1$ which contradicts $k > n+2$.

($k = t_1$) This is when R_i receives first attention with tickets t_0, t_1, t_2 (see Figure 4). $\mathcal{C}_n$ must occur somewhere at the highlighted positions in Figure 4 because this fragment of the structure is copied leading to an increase of c_f. Hence, $n = t_0$ or $n = t_1$, or $\mathcal{C}_n$ occurs in $\mathcal{T}$. $n \neq t_0$ because otherwise $n = t_0$, $k = t_1 = t_0 + 1 = n + 1$ which contradicts $k > n + 2$. n cannot be t_1 because otherwise $n = t_1 = k$ which contradicts $k > n + 2$. Hence, $\mathcal{C}_n$ occurs in $\mathcal{T}$. However, this is also not possible for the following reason. We know that $k = t_1$ enters I so this is due to R_i acting when receiving the first attention with tickets t_0, t_1, t_2. This means that no requirement R_j of higher priority than R_i (i.e., with $j < i$) has received attention after R_i got associated with tickets t_0, t_1, t_2 (up to the current stage) – otherwise R_i's tickets would have been reassigned to numbers different than t_0, t_1, t_2. Therefore, $\mathcal{C}_n$ entered the construction *after* R_i was assigned to t_0, t_1, t_2. Hence, by the construction (i.e. the way we choose and assign tickets to requirements (re)entering the construction), n is a ticket for some lower priority requirement R_l ($l > i$). But when n enters the construction as a ticket of such R_l, n is chosen as a fresh number so, in particular, $n > t_1 = k$ which contradicts $k > n + 2$.

($k = t_2$) This is when $\mathcal{R}_i$ receives attention for the second time with tickets t_0, t_1, t_2 (see Figure 6). $\mathcal{C}_n$ occurs somewhere at the highlighted positions in Figure 6, i.e. $n = t_0$ or $n = t_1$, or $\mathcal{C}_n$ occurs in $\mathcal{T} + \mathcal{U}$. $n \neq t_0$ because otherwise $n = t_0$, $k = t_2 = t_0 + 2 = n + 2$ which contradicts $k > n + 2$. $n \neq t_1$ because otherwise $n = t_1$, $k = t_2 = t_1 + 1 = n + 1$ which contradicts $k > n + 2$. Therefore, n occurs in $\mathcal{T} + \mathcal{U}$. The rest of the argument is similar to the analogical place of the previous case ($k = t_1$). ◀

▶ **Lemma 27.** $c_f \leq_T I$.

Proof. To compute $c_f(n)$, find s such that $I_s[n + 2] = I[n + 2]$. By Lemma 26 and the fact that $c_f(n)$ is increased *only* due to numbers entering I, $c_f(n)$ is not increased at stages $> s$ (no additional copies of $\mathcal{C}_n$ are added to $f_{\mathcal{A}}$). Return the number of copies of $\mathcal{C}_n$ in $f_{\mathcal{A}_s}$. ◀

Due to space constraints, the proof of the following lemma can be found in the full version.

▶ **Lemma 28.** $f_{\mathcal{A}} \leq_T c_f$.

By Lemmas 25, 26, 27 and 28: $\mathbf{0} <_T f_{\mathcal{A}} \leq c_f \leq_T I <_T \mathbf{0}'$. The spectrum of f is not trivial by Proposition 6. By Theorem 1, $DgSp(f)$ contains all c.e. degrees. Since $f_{\mathcal{A}}$ is of non-c.e. degree, $DgSp(f) \neq$ the c.e. degrees. To show that $DgSp(f) \neq$ the Δ_2 degrees, we need the following theorem.

▶ **Theorem 29** (Cooper, Lempp and Watson, [4]). *Given c.e. sets $U <_T V$ there is a proper d.c.e. set C of properly d.c.e. degree such that $U <_T C <_T V$.*

Assume, for a contradiction, that $DgSp(f)$ consists of the Δ_2 degrees. By Theorem 29, $DgSp(f) \cap \{\deg(A) : c_f \leq_T A \leq_T \mathbf{0}'\}$ contains a proper d.c.e. degree. However, by Proposition 22, $DgSp(f) \cap \{\deg(A) : c_f \leq_T A \leq_T \mathbf{0}'\}$ contains only c.e. degrees. This is a contradiction, so the degree spectrum of f is different then the Δ_2 degrees. This completes the proof.

4 Conclusions and Open Questions

We have investigated the problem of intrinsic complexity of computable relations on $(\omega, <)$, as measured by their degree spectra, in the restricted setting of graphs of unary total computable functions. It has been known that possible candidates for intrinsic complexities of such functions include three sets consisting of precisely: the computable degree, all c.e.

degrees, and all Δ_2 degrees. Imposing certain structural constraints on such functions has led us to the notions of block functions (Definition 10) and a broader class of quasi-block functions (Definition 15). Non-quasi-block functions have intrinsic complexity equal to the c.e. degrees (Theorem 18) which redirects all focus to quasi-block functions. We have obtained several results on this class, most prominently the one on block-functions with finitely many types (Theorem 14) showing that their intrinsic complexity is either trivial or equal to the Δ_2 degrees. However, the most surprising result is that on an unusual degree spectrum (Theorem 23) which proves the existence of a block function having intrinsic complexity different from the already known three candidates. To the best of our knowledge, this theorem answers Question 6.2 from [21] formulated by Wright who asked whether there are relations on $(\omega, <)$ with other degree spectra (than the three known candidates). Harrison-Trainor obtained a related result though for a different relation. However, for his relation it is not known whether its spectrum is intermediate (see Section 1 for details, as well as [10]).

A few questions arise immediately. Note, for example, that our solution to Wright's question invites the hypothesis, possibly to be proven using some kind of permitting, that there exist infinitely many spectra of computable block functions on $(\omega, <)$. A parallel question is what degrees such nonstandard spectra contain. Observe that even for the function constructed in Theorem 23 the exact contents of its spectrum are unknown. We finish the paper with the general open question: what are the possible kinds of nonstandard spectra of computable block functions on $(\omega, <)$?

References

1 Chris J. Ash and Julia Knight. *Computable structures and the hyperarithmetical hierarchy*, volume 144 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam, 2000.

2 Jennifer Chubb, Andrey Frolov, and Valentina Harizanov. Degree spectra of the successor relation of computable linear orderings. *Archive for Mathematical Logic*, 48(1):7–13, 2009. doi:10.1007/s00153-008-0110-6.

3 S. Barry Cooper. *Degrees of unsolvability.* PhD thesis, University of Leicester, 1971.

4 S. Barry Cooper, Steffen Lempp, and Philip Watson. Weak density and cupping in the d-r.e. degrees. *Israel Journal of Mathematics*, 67(2):137–152, 1989. doi:10.1007/BF02937291.

5 Rod Downey, Bakhadyr Khoussainov, Joseph S. Miller, and Liang Yu. Degree spectra of unary relations on $(\omega, \leq)$. In *Logic, Methodology and Philosophy of Science: Proceedings of the Thirteenth International Congress*, pages 35–55. College Publications, 2009. Publisher: Citeseer. URL: http://homepages.mcs.vuw.ac.nz/~downey/publications/LOJan24.pdf.

6 Richard L. Epstein. *Degrees of Unsolvability: Structure and Theory*, volume 759 of *Lecture Notes in Mathematics.* Springer, Berlin Heidelberg, 1979.

7 Ekaterina B. Fokina, Valentina Harizanov, and Alexander Melnikov. Computable model theory. In R. Downey, editor, *Turing's legacy: Developments from Turing's ideas in logic*, volume 42 of *Lecture Notes in Logic*, pages 124–194. Cambridge University Press, Cambridge, 2014. doi:10.1017/CBO9781107338579.006.

8 E. Mark Gold. Limiting Recursion. *Journal of Symbolic Logic*, 30(1):28–48, 1965. doi:10.2307/2270580.

9 Valentina S. Harizanov. *Degree spectrum of a recursive relation on a recursive structure.* PhD thesis, University of Wisconsin-Madinson, 1987.

10 Matthew Harrison-Trainor. Degree spectra of relations on a cone. *Memoirs of the American Mathematical Society*, 253(1208):1–120, 2018. doi:10.1090/memo/1208.

11 Denis R. Hirschfeldt. Degree spectra of relations on computable structures. *Bulletin of Symbolic Logic*, 6(2):197–212, 2000. doi:10.2307/421207.

12 Carolyn Alexis Knoll. Degree spectra of unary relations on ω and ζ. Master's thesis, University of Waterloo, 2009.

13 Dragoslav S. Mitrinović, József Sándor, and Borislav Crstici. *Handbook of Number Theory*, volume 351 of *Mathematics and Its Applications*. Kluwer, 1995.

14 Antonio Montalbán. *Computable structure theory: Within the arithmetic*. Cambridge University Press, 2021.

15 Michael Moses. Relations Intrinsically Recursive in Linear Orders. *Mathematical Logic Quarterly*, 32(25-30):467–472, 1986. doi:10.1002/malq.19860322514.

16 Hilary Putnam. Trial and Error Predicates and the Solution to a Problem of Mostowski. *Journal of Symbolic Logic*, 30(1):49–57, 1965. doi:10.2307/2270581.

17 Linda Jean Richter. Degrees of structures. *Journal of Symbolic Logic*, 46(4):723–731, 1981. Publisher: Cambridge University Press. doi:10.2307/2273222.

18 Stewart Shapiro. Acceptable notation. *Notre Dame Journal of Formal Logic*, 23(1):14–20, January 1982. doi:10.1305/ndjfl/1093883561.

19 Joseph R. Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, 69:644–653, 1959. doi:10.2307/1970028.

20 Robert I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, New York, NY, USA, 1987.

21 Matthew Wright. Degrees of relations on ordinals. *Computability*, 7(4):349–365, 2018. doi:10.3233/COM-180086.

A Proof of Theorem 14

▶ **Theorem 14.** *Let f be a computable block function such that it has only finitely many f-types and f is not almost identity. Then the spectrum $DgSp(f)$ consists of all Δ_2 degrees.*

Proof. Let $I_0, I_1, \dots, I_N$ be all (isomorphism types of) f-blocks. We represent the structure $\mathcal{B} = (\omega, <, f)$ by α_f according to Remark 11. As in Theorem 8, we fix a Δ_2 set X. Our goal is to construct a computable copy $\mathcal{A} = (\omega, <_{\mathcal{A}})$ of $(\omega, <)$ such that $f_{\mathcal{A}} \equiv_T X$. In general, we follow the notations of Theorem 8 (e.g., $f_{\mathcal{A}_s}(x)$ is defined in the same way as in the previous proof).

Beforehand, we choose a large enough number M such that:

- M lies at the right end of its f-block (inside $\mathcal{B}$),
- for every $x > M$, the isomorphism type of its f-block occurs infinitely often in $\mathcal{B}$.

As in the proof of Theorem 8, we copy the interval $[0; M]$ into our structure $\mathcal{A}$, and all new elements will be added to the right of this interval.

The proof is split into three cases which depend on the properties of the string α_f (each of the cases requires a separate construction):

(a) There are two different finite strings σ and τ such that:
- the lengths of σ and τ are the same;
- τ can be obtained via a permutation of σ, i.e., there is a permutation h of the set $\{0, 1, \dots, |\sigma| - 1\}$ such that $\tau(i) = \sigma(h(i))$, for all $i < |\sigma|$;
- both σ and τ occur infinitely often in α_f.

(b) There is only one block I_k such that k occurs infinitely often in α_f.

(c) Neither of the previous two cases holds.

Case (a). For the sake of simplicity, we give a detailed proof for the case when $\sigma = 01$ and $\tau = 10$. After that, we explain how to deal with the general case.

Our construction satisfies the following requirements:

$$\begin{aligned} e \in X &\Leftrightarrow 2e \text{ belongs to a block isomorphic to } I_1, \\ e \notin X &\Leftrightarrow 2e \text{ belongs to a block isomorphic to } I_0. \end{aligned} \tag{$\mathcal{R}_e$}$$

Suppose that $|I_0| + |I_1| = q + 1$.

Strategy $\mathcal{R}_e$ in isolation. When $\mathcal{R}_e$ starts working at a stage $s_0 + 1$, we proceed as follows. Assume that $X_{s_0}(e) = 1$ (the other case is treated similarly). We choose q fresh odd numbers $c_1^e, c_2^e, \dots, c_q^e$ and declare them the *companions* of $2e$. We add the chain

$$2e <_{\mathcal{A}} c_1^e <_{\mathcal{A}} c_2^e <_{\mathcal{A}} \dots <_{\mathcal{A}} c_q^e$$

to the right of $\mathcal{A}_{s_0}$. If needed, we add finitely many fresh odd numbers in-between $\mathcal{A}_{s_0}$ and $2e$. This procedure ensures that (at the moment) the finite structure $([2e; c_q^e]_{\mathcal{A}}, <_{\mathcal{A}}, f_{\mathcal{A}})$ is isomorphic to the disjoint sum $I_1 + I_0$.

The strategy $\mathcal{R}_e$ *requires attention* at a stage s if inside the current $\mathcal{A}_s$, the number $2e$ belongs to a copy of $I_{1-X_s(e)}$. When $\mathcal{R}_e$ requires attention, we apply a PtR-module.

The PtR-module for $\mathcal{R}_e$. We choose the middle interval C as the set containing $2e$ and all its companions, i.e. $C = \{2e <_{\mathcal{A}} c_1^e <_{\mathcal{A}} \dots <_{\mathcal{A}} c_q^e\}$. Our target condition is defined as follows: inside the resulting structure $\mathcal{F}$, the structure $(C, <_{\mathcal{F}}, f_{\mathcal{F}} \upharpoonright C)$ is isomorphic to the disjoint sum $I_{X_s(e)} + I_{1-X_s(e)}$. As in Theorem 8, the structure $\mathcal{F}$ is treated as output of the module.

The construction is arranged similarly to that of Theorem 8.

Verification. We need to show that every application of a PtR-module is successful. This follows from two observations:

1. If we want to “transform”, say, $I_0 + I_1$ into $I_1 + I_0$, then this can be achieved by an appropriate pushing to the right, since the string $\tau = 10$ occurs infinitely often in α_f.
2. Remark 9 guarantees that one can also safely push the interval D (from the PtR-module): notice that if some block I_r occurs in D, then r occurs infinitely often in α_f.

Since pushing to the right is always successful, every requirement $\mathcal{R}_e$ is satisfied. Note that given $f_{\mathcal{A}}$ as an oracle, one can recover the $f_{\mathcal{A}}$-block of $2e$. This fact (together with $\mathcal{R}_e$-requirements) implies that $X \leq_T f_{\mathcal{A}}$.

Every element $a \in \mathcal{A}$ is pushed to the right only finitely often. Therefore, the structure $\mathcal{A}$ is a computable copy of $(\omega, <)$.

Given an odd number $x = 2k + 1$, one can computably determine which of the following two cases holds:

1. $2k + 1$ is a companion c_t^e of some even number $2e$ (in this case, the indices e and t are also computed effectively), or
2. $2k + 1$ is added as a “filler” by some action of an $\mathcal{R}_e$-strategy (either by its initial actions, or by an application of a PtR-module).

In the second case, the value $f_{\mathcal{A}_s}(x)$ never changes (after being defined for the first time). In the first case, the oracle X can tell us whether $x = c_t^e$ belongs to (a copy of) I_0 or I_1, and X can also compute the image $f_{\mathcal{A}}(x)$. In a similar way, X computes the images $f_{\mathcal{A}}(2e)$, for $e \in \omega$. Hence, we obtain that $f_{\mathcal{A}} \equiv_T X$. This concludes the case when $\sigma = 01$ and $\tau = 10$.

The case of arbitrary σ and τ follows a similar proof outline. We illustrate this by considering $\sigma = 012301$ and $\tau = 013021$. Then our construction will switch between finite structures

$$\mathcal{F}_\sigma = I_0 + I_1 + I_2 + I_3 + I_0 + I_1 \text{ and } \mathcal{F}_\tau = I_0 + I_1 + I_3 + I_0 + I_2 + I_1.$$

Since both σ and τ occur infinitely often in α_f, an appropriate PtR-module can always "transform" $\mathcal{F}_\sigma$ into $\mathcal{F}_\tau$, and vice versa.

During the construction, an even number $2e$ will always belong to the third block from the left inside $\mathcal{F}_\square$ (i.e., either I_2 in $\mathcal{F}_\sigma$, or I_3 in $\mathcal{F}_\tau$). The third block is chosen because it corresponds to the first position, where σ and τ differ. The rest of the corresponding copy of $\mathcal{F}_\square$ consists of companions of $2e$. In the final structure $\mathcal{A}$, we will achieve the following: if $e \in X$, then $2e$ lies in a copy of I_2; otherwise, $2e$ belongs to a copy of I_3. This concludes the discussion of Case (a).

Case (b). Without loss of generality, we assume that $I_k = I_0$. We satisfy the following requirements:

$$\begin{aligned} e \in X &\iff 2e \text{ lies at the right end of a copy of } I_0,\\ e \notin X &\iff 2e \text{ lies at the left end of a copy of } I_0. \end{aligned} \tag{$\mathcal{R}_e$}$$

Suppose that $|I_0| = q+1$. Notice that $q \geq 1$, since f is not almost identity.

Strategy $\mathcal{R}_e$ in isolation. $2e$ will have finitely many odd numbers as its *companions*. In contrast to Case (a), these companions could be added stage-by-stage.

When $\mathcal{R}_e$ starts working at a stage s_0+1, we proceed as follows. Suppose $X_{s_0}(e) = 1$ (the other case is similar). Then we choose q fresh odd numbers $c_1, \dots, c_q$, and declare that they are companions of $2e$. We set $c_1 <_\mathcal{A} \dots <_\mathcal{A} c_q <_\mathcal{A} 2e$ (these elements are added to the right of $\mathcal{A}_{s_0}$). We ensure that the structure $([c_1; 2e]_\mathcal{A}, <_\mathcal{A}, f_\mathcal{A})$ is isomorphic to I_0 (if needed, one adds fresh odd numbers in-between $\mathcal{A}_{s_0}$ and c_1).

We also ensure that by the end of each stage s, $2e$ and its (current) companions form an interval inside $\mathcal{A}_s$, and this interval can be treated as a sum of blocks (in $\mathcal{A}_s$).

The strategy $\mathcal{R}_e$ *requires attention* at a stage s if inside the current $\mathcal{A}_s$, the corresponding requirement is not satisfied (e.g., if $X_s(e) = 0$ and $2e$ lies at the right end of I_0). When $\mathcal{R}_e$ requires attention, we apply a PtR-module.

The PtR-module for $\mathcal{R}_e$. We choose the middle interval C as the set containing $2e$ and all its current companions. We consider the following two subcases.

Subcase 1. Assume that right now, $X_s(e) = 1$ and $2e$ lies at the left end of a copy of I_0.
Then our target condition is defined as follows: inside the resulting output structure $\mathcal{F}$, the number $2e$ should belong to the right end of a copy of I_0.
In order to achieve this condition, we add precisely q fresh odd numbers in-between B and C, and only one fresh odd number in-between C and D. This guarantees that $2e$ "moves" to the right end of a block.

Subcase 2. Otherwise, suppose that $X_s(e) = 0$ and $2e$ lies at the right end of a copy of I_0.
Then we pursue the following condition: inside the output $\mathcal{F}$, $2e$ should "move" to the left end of a block I_0.
In order to do this, we add one fresh number in-between B and C, and q fresh numbers in-between C and D.
In both subcases, we declare that the newly added odd numbers belong to the set of companions of $2e$.

The construction is arranged similarly to the previous ones.

Verification. Since almost every block from α_f is isomorphic to I_0, every application of a PtR-module is successful. In addition, the actions of the PtR-module for $\mathcal{R}_e$ does not injure other strategies.

We deduce that all requirements $\mathcal{R}_e$ are satisfied. Given $f_{\mathcal{A}}$ as an oracle, one can recover the position of $2e$ inside its $f_{\mathcal{A}}$-block. This implies that $X \leq_T f_{\mathcal{A}}$. In addition, a standard argument shows that $\mathcal{A}$ is a computable copy of $(\omega, <)$.

Notice the following. Since $2e$ and its companions always stick together as an interval, there are only two possible variants of the final $f_{\mathcal{A}}$-block of $2e$: either it contains q companions of $2e$ added at the very beginning of the work of the $\mathcal{R}_e$-strategy, or it contains q closest (inside $\mathcal{A}$) companions of $2e$ added by the first application of the PtR-module for $\mathcal{R}_e$.

As in the previous case, given an odd number $x = 2k + 1$, one can determine which of the following two cases holds:

1. x is a companion of some even number $2e$ (the index e is recovered effectively), or
2. x is added as a "filler" by some action of an $\mathcal{R}_e$-strategy.

In the second case, the value $f_{\mathcal{A}_s}(x)$ never changes. In the first case, the oracle X can tell us the content of the final $f_{\mathcal{A}}$-block containing x: indeed, if $X_{s_0}(e) = X_{s_1}(e)$, then at the stages s_0 and s_1, the blocks of x inside $\mathcal{A}_{s_0}$ and $\mathcal{A}_{s_1}$ contain precisely the same elements. We deduce that $f_{\mathcal{A}} \leq_T X$. This concludes the proof of Case (b).

Case (c). Before describing the construction, we provide a combinatorial analysis of the string α_f.

▶ **Lemma 30.** *If the string α_f satisfies neither Case (a) nor Case (b), then there are symbols $b, d, e \in \Sigma$ such that $d \neq b$, $e \neq b$, and for every natural number n, there exists $m > n$ such that the finite string $db^m e$ occurs in α_f.*

Proof. Without loss of generality, one may assume that every symbol from Σ occurs infinitely often in α_f.

For a finite string σ over the alphabet Σ, we denote

$$\#(\sigma) = |\{a \in \Sigma : a \text{ occurs in } \sigma\}|.$$

We choose a finite string τ such that τ occurs infinitely often in α_f and

$$\#(\tau) = \max\{\#(\sigma) : \sigma \text{ occurs infinitely often in } \alpha_f\}. \tag{1}$$

Let c be the last symbol of the string τ.

There exists a symbol b such that the string $\tau_b = \tau\, b$ occurs infinitely often in α_f. Equation (1) implies that b occurs in τ (indeed, if b does not occur in τ, then $\#(\tau_b) = \#(\tau)+1$).

We prove that $c = b$. Towards a contradiction, assume that $c \neq b$. Then τ can be decomposed as $\tau = \xi\, b\, \delta\, c^k$ for some $k \geq 1$ and finite strings ξ, δ. The string $\tau_b = \xi\, b\, \delta\, c^k\, b$ occurs infinitely often in α_f. In turn, this implies that both $b\, \delta\, c^k$ and $\delta\, c^k\, b$ occur infinitely often in α_f. Therefore, α_f satisfies Case (a), which gives a contradiction.

Hence, we have $\tau = \rho\, b^k$ for some $k \geq 1$ and finite string ρ, and the string $\tau_b = \rho\, b^{k+1}$ occurs infinitely often in α_f. Note that $\#(\tau_b) = \#(\tau)$. This implies that by applying induction, one can show that for *every* $l \geq 1$,

$$\rho\, b^l \text{ occurs infinitely often in } \alpha_f. \tag{2}$$

Since α_f does not satisfy Case (b), there are at least two different symbols occuring infinitely often in α_f. This fact and Equation (2) imply that for every $n \in \omega$, there exist $m > n$ and two symbols d' and e' such that $d' \neq b$, $e' \neq b$, and $d' b^m e'$ occurs in α_f. After that, we apply the pigeonhole principle to finish the proof of the lemma. ◀

By Lemma 30, we may assume that for every $n \in \omega$, there exists $m > n$ such that, say, $10^m 2$ occurs in α_f. We satisfy the same requirements as in Case (b):

$$\begin{aligned} e \in X &\;\Leftrightarrow\; 2e \text{ lies at the right end of a copy of } I_0, \\ e \notin X &\;\Leftrightarrow\; 2e \text{ lies at the left end of a copy of } I_0. \end{aligned} \qquad (\mathcal{R}_e)$$

In general, our notations also follow those of Case (b).

Strategy $\mathcal{R}_e$ in isolation. When $\mathcal{R}_e$ starts working at a stage $s_0 + 1$, we proceed as follows. Suppose $X_{s_0}(e) = 0$. We find a large enough number m such that $10^m 2$ occurs in α_f, and the corresponding sequence of f-blocks $I_1 + I_0 + I_0 + \ldots + I_0 + I_2$ does not intersect with the image of $\mathcal{A}_{s_0}$ inside $(\omega, <)$.

We add $2e$ and fresh odd numbers into $\mathcal{A}$ ensuring that the newly added elements form a sequence of $f_{\mathcal{A}}$-blocks:

$$I_1 + \underbrace{I_0 + \ldots + I_0}_{m \text{ times}} + I_2;$$

if needed, fresh odd numbers are also added in-between $\mathcal{A}_{s_0}$ and this sequence. The number $2e$ lies at the left end of the leftmost block I_0. The elements forming I_1 and I_2 are declared *boundary companions* of $2e$. The odd numbers forming the inner sequence of I_0-s are declared *non-boundary companions* of $2e$.

As usual, $\mathcal{R}_e$ *requires attention* at a stage s if inside the current $\mathcal{A}_s$, the corresponding requirement is not satisfied. When $\mathcal{R}_e$ requires attention, we apply a PtR-module.

The PtR-module for $\mathcal{R}_e$. We choose the middle interval C as the set containing $2e$ and all its companions. Assume that right now, $X_s(e) = 0$ and $2e$ lies at the right end of a copy of I_0 (the other subcase is treated in a similar way). Then the target condition is defined as follows: inside $\mathcal{F}$, the number $2e$ belongs to the left end of a copy of I_0.

Suppose that right now, the companions of $2e$ form a sequence of $f_{\mathcal{A}_s}$-blocks corresponding to a finite string $10^m 2$.

We always assume the following: if a fresh number x is added between some companions of some $2j$, then it is declared a non-boundary companion of $2j$. In addition, every such x is put between the I_1-block and the I_2-block containing the boundary companions of $2j$. Moreover, we require that inside the resulting structure $\mathcal{F}$, the element x becomes a part of a copy of I_0.

In order to achieve the target condition, we proceed as follows. First, we find a large enough $m' > m$ such that $10^{m'} 2$ occurs in α_f, and this occurrence of $10^{m'} 2$ lies to the right of the image of $\mathcal{A}_s$ inside $(\omega, <)$. We add fresh odd numbers in such a way that:

- The companions of $2e$ (including newly added companions) form a sequence of $f_{\mathcal{F}}$-blocks corresponding to $10^{m'} 2$ (inside α_f). This is achieved by adding numbers in-between B and C, and by adding fresh I_0-blocks between the I_1-block and the I_2-block containing the boundary companions of $2e$.
- Similarly to Case (b), this procedure must ensure that $2e$ moves to the left end of an I_0-block.

Second, we carefully push the companions of $2j$, where $e < j < s$, to the right. Consider each such j (in turn). Suppose that the companions of $2j$ form a sequence of $f_{\mathcal{A}_s}$-blocks corresponding to a finite string $10^{m_j} 2$. We choose a large enough $m'_j > m_j$ (again, with $10^{m'_j} 2$ occuring in α_f to the right of the image of the current (preliminary) version of $\mathcal{F}$). We add fresh numbers in such a way that:

- The companions of $2j$ (including new ones) form a sequence of $f_{\mathcal{F}}$-blocks corresponding to $10^{m'_j}2$ inside α_f.
- If x is a new companion of $2j$, then it belongs to a new I_0-block which corresponds to one of the underlined zeros in the following decomposition:

$$10^{m'_j}2 = 10^{m_j}\underline{0}\underline{0}\ldots\underline{0}2.$$

This careful pushing allows to ensure that the PtR-module does not injure strategies $\mathcal{R}_j$, for $j \neq e$. Indeed, after the pushing, the value $f_{\mathcal{A}}(2j)$ does not change.

The construction is arranged in a similar way as before.

Verification. The fact that α_f contains occurrences of 10^m2 for arbitrarily large m implies that every application of a PtR-module is successful. We deduce that all requirements $\mathcal{R}_e$ are satisfied. The rest of the verification is similar to that of Case (b). This concludes the proof of Theorem 14. ◀

Subquadratic-Time Algorithm for the Diameter and All Eccentricities on Median Graphs

Pierre Bergé[1]
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, France
IRIF, CNRS, Université de Paris, France

Guillaume Ducoffe
National Institute of Research and Development in Informatics, Bucharest, Romania
University of Bucharest, Romania

Michel Habib
IRIF, CNRS; Université de Paris Cité, France

Abstract

On sparse graphs, Roditty and Williams [2013] proved that no $O(n^{2-\varepsilon})$-time algorithm achieves an approximation factor smaller than $\frac{3}{2}$ for the diameter problem unless SETH fails. We answer here an open question formulated in the literature: can we use the structural properties of median graphs to break this global quadratic barrier?

We propose the first combinatorial algorithm computing exactly all eccentricities of a median graph in truly subquadratic time. Median graphs constitute the family of graphs which is the most studied in metric graph theory because their structure represents many other discrete and geometric concepts, such as CAT(0) cube complexes. Our result generalizes a recent one, stating that there is a linear-time algorithm for computing all eccentricities in median graphs with bounded dimension d, *i.e.* the dimension of the largest induced hypercube (note that 1-dimensional median graphs are exactly the forests). This prerequisite on d is not necessarily anymore to determine all eccentricities in subquadratic time. The execution time of our algorithm is $O(n^{1.6456} \log^{O(1)} n)$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Data structures design and analysis

Keywords and phrases Diameter, Eccentricities, Metric graph theory, Median graphs, Hypercubes

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.9

Related Version *Full Version*: `https://arxiv.org/abs/2110.02709`

1 Introduction

Median graphs can be certainly identified as the most important family of graphs in metric graph theory. They are related to numerous areas: universal algebra [3, 16], CAT(0) cube complexes [5, 20], abstract models of concurrency [11, 37], and genetics [8, 10]. Let $d(a, b)$ be the length (*i.e.* the number of edges) of the shortest (a, b)-path for $a, b \in V$ and $I(a, b)$ be the set made up of all vertices u metrically between a and b, *i.e.* $d(a, b) = d(a, u) + d(u, b)$. Median graphs are the graphs such that for any triplet of distinct vertices $x, y, z \in V$, set $I(x, y) \cap I(y, z) \cap I(z, x)$ is a singleton, containing the *median* $m(x, y, z)$ of this triplet.

The purpose of this article is to break the quadratic barrier for the computation time of certain metric parameters on median graphs. In particular, we focus on one of the most fundamental problems in algorithmic graph theory related to distances: the *diameter*. Given an undirected graph $G = (V, E)$, the diameter is the maximum distance $d(u, v)$ for all $u, v \in V$. Two vertices at maximum distance form a *diametral pair*. An even more general

[1] Corresponding author

problem consists in determining all eccentricities of the graph. The eccentricity $\mathsf{ecc}(v)$ of a vertex v is the maximum length of a shortest path starting from v: $\mathsf{ecc}(v) = \max_{w \in V} d(v, w)$. The diameter is thus the maximum eccentricity.

1.1 State of the art

Executing a *Breadth First Search* (BFS) from each vertex of an input graph G suffices to obtain its eccentricities in $O(n\,|E|)$, with $n = |V|$. As median graphs are relatively sparse, $|E| \leq n \log n$, these multiple BFSs compute all eccentricities in time $O(n^2 \log n)$ for this class of graphs. Very efficient algorithms determining the diameter already exist on other classes of graphs, for example [2, 18, 24]. Many works have also been devoted to approximation algorithms for this parameter. Chechik *et al.* [19] showed that the diameter can be approximated within a factor $\frac{3}{2}$ in time $\tilde{O}(m^{\frac{3}{2}})$ on general graphs. On sparse graphs, it was shown in [36] that no $O(n^{2-\varepsilon})$-time algorithm achieves an approximation factor smaller than $\frac{3}{2}$ for the diameter unless the Strong Exponential Time Hypothesis (SETH) fails.

Median graphs are bipartite and can be isometrically embedded into hypercubes. They are the 1-skeletons of CAT(0) cube complexes [20] and the domains of event structures [11]. They admit structural properties, such as the Mulder's convex expansion [34, 35]. They are strongly related to hypercubes retracts [4], Cartesian products and gated amalgams [5], but also Helly hypergraphs [33]. They do not contain induced $K_{2,3}$, otherwise a triplet of vertices would admit at least two medians. The *dimension* d of a median graph G is the dimension of its largest induced hypercube. The value of this parameter is at most $\lfloor \log n \rfloor$ and meets this upper bound when G is a hypercube. Moreover, parameter d takes part in the sparsity of median graphs: $|E| \leq dn$.

An important concept related to median graphs is the equivalence relation Θ. This is the reflexive and transitive closure of relation Θ_0, where two edges are in Θ_0 if they are opposite in a common 4-cycle. A Θ-*class* is an equivalence class of Θ. Each Θ-class of a median graph forms a matching cutset, splitting the graph into two convex connected components, called *halfspaces*. The number $q \leq n$ of Θ-classes corresponds to the dimension of the hypercube in which the median graph G isometrically embeds. Value q satisfies the Euler-type formula $2n - m - q \leq 2$ [29]. A recent LexBFS-based algorithm [13] identifies the Θ-classes in linear time $O(|E|) = O(dn)$.

There exist efficient algorithms for some metric parameters on median graphs. For example, the median set and the Wiener index can be determined in $O(|E|)$ [13]. Subfamilies of median graphs have also been studied. There is an algorithm computing the diameter and the radius in linear time for squaregraphs [21]. A more recent contribution introduces a quasilinear time algorithm - running in $O(n \log^{O(1)} n)$ - for the diameter on cube-free median graphs [23], using distance and routing labeling schemes proposed in [22]. Eventually, a linear-time algorithm [15] for the diameter on constant-dimension median graphs was proposed, *i.e.* for median graphs satisfying $d = O(1)$.

The existence of a truly subquadratic-time algorithm for the diameter on all median graphs is open and was recently formulated in [13, 15, 23]. An even more ambitious question can be asked. Can this subquadratic barrier be overpassed for the problem of finding all eccentricities of a median graph ? As the total size of the output is linear and this problem generalizes the diameter one, this question is legitimate. More generally, the question holds for all metric parameters (except the median set and the Wiener index for which a linear-time algorithm was recently designed). We propose here the first subquadratic-time algorithm computing all eccentricities on median graphs.

1.2 Contributions

Our first contribution in this paper is the design of a quasilinear, *i.e.* $O((\log n)^{O(1)}n)$, time algorithm computing the diameter of simplex graphs. A simplex graph $K(G) = (V_K, E_K)$ of a graph G is obtained by considering the induced complete graphs (cliques) of G as vertices V_K. Then, two of these cliques are connected by an edge if they differ by only one element: one is C, the other is $C \cup \{v\}$. These edges form the set E_K. All simplex graphs are median [5, 12]. Moreover, we observe that simplex graphs fulfil an interesting property: they admit a central vertex - representing the empty clique - and every Θ-class has an edge incident to that vertex.

First, this algorithm extends the set of median graphs for which a quasilinear time procedure computing the diameter exists. Indeed, simplex graphs form a sub-class of median graphs containing instances with unbounded dimension d.

- There is a combinatorial algorithm determining the diameter and all eccentricities of simplex graphs in $O((d^3 + \log n)n)$: Theorem 3.8, Section 3.

Second, we remark that this method can be integrated to the algorithm already proposed in [15] to compute all eccentricities of median graphs in time $O(2^{O(d \log d)}n)$. This allows us to decrease this running time. Thanks to this modification, the new algorithm proposed computes all eccentricities of a median graph in $\tilde{O}(2^{2d}n)$, where notation $\tilde{O}$ neglects polylogarithmic factors. Even if the algorithm stays linear for constant-dimension median graphs, observe that the dependence on d decreases, from a slightly super-exponential function to a simple exponential one.

- There is a combinatorial algorithm determining all eccentricities of median graphs in $\tilde{O}(2^{2d}n)$: Theorem 4.16, Section 4.1.

The second and main contribution in this paper is the design of a subquadratic-time dynamic programming procedure which computes all eccentricities of any median graph. Here, the linear-time simple-exponential-FPT algorithm for all eccentricities presented above plays a crucial role: it is the base case. This framework consists in partitioning recursively the input graph G into the halfspaces of its largest Θ-class. With our construction, the leaves of this recursive tree are median graphs with dimension at most $\lfloor \frac{1}{3} \log n \rfloor$ and we can apply the former linear-time FPT algorithm.

- There is a combinatorial algorithm determining all eccentricities of median graphs in $\tilde{O}(n^{\frac{5}{3}})$: Theorem 4.22, Section 4.2.

We terminate by mentioning a possible improvement of this algorithm. Based on a faster enumeration of sets of pairwise orthogonal Θ-classes, its running time can be decreased to $\tilde{O}(n^{\beta})$, where $\beta = 1.6456$. Due to page limit, this extra part is omitted. A long version of this paper presents this result [14].

All these outcomes put in evidence a relationship between the design of linear-time FPT algorithms and the design of subquadratic-time algorithms determining metric parameters on median graphs. We hope that the ideas proposed to establish all these results represent interesting tools to break the quadratic barrier on other open questions.

1.3 Organization

In Section 2, we remind the definition of median graphs. The well-known properties and concepts related to them are listed, among them Θ-classes, signature, and POFs. Section 3 summarizes our contribution on simplex graphs. In Section 4, we show how to obtain a linear-time simple-exponential-FPT algorithm for all eccentricities of a median graph, parameterized by the dimension d. Thanks to it, we propose a dynamic programming procedure to reduce the computation of eccentricities of any median graph to the same problem on a collection of median subgraphs of sub-logarithmic dimension.

2 Median graphs

In this section, we recall some notions related to distances in graphs, and more particularly median graphs. Two important tools are presented: the Θ-classes, which are equivalence classes over the edge set, and the *Pairwise Orthogonal Families* (POFs) characterizing Θ-classes belonging to a common hypercube.

2.1 Θ-classes

All graphs $G = (V, E)$ considered in this paper are undirected, unweighted, simple, finite and connected. We denote by $N(u)$ the *open neighborhood* of $u \in V$, *i.e.* the set of vertices adjacent to u in G. We extend it naturally: for any set $A \subseteq V$, the neighborhood $N(A)$ of A is the set of vertices outside A adjacent to some $u \in A$.

Given two vertices $u, v \in V$, let $d(u, v)$ be the *distance* between u and v, *i.e.* the length of the shortest (u, v)-path. The *eccentricity* $\mathsf{ecc}(u)$ of a vertex $u \in V$ is the length of the longest shortest path starting from u. Put formally, $\mathsf{ecc}(u)$ is the maximum value $d(u, v)$ for all $v \in V$: $\mathsf{ecc}(u) = \max_{v \in V} d(u, v)$. The diameter of graph G is the maximum distance between two of its vertices: $\mathsf{diam}(G) = \max_{u \in V} \mathsf{ecc}(u)$.

We denote by $I(u, v)$ the *interval* of pair u, v. It contains exactly the vertices which lie metrically between u and v: $I(u, v) = \{x \in V : d(u, x) + d(x, v) = d(u, v)\}$. The vertices of $I(u, v)$ are lying on at least one shortest (u, v)-path.

We say that a set $H \subseteq V$ (or the induced subgraph $G[H]$) is *convex* if $I(u, v) \subseteq H$ for any pair $u, v \in H$. Moreover, we say that H is *gated* if any vertex $v \notin H$ admits a *gate* $g_H(v) \in H$, *i.e.* a vertex that belongs to all intervals $I(v, x)$, $x \in H$. For any $x \in H$, we have $d(v, g_H(v)) + d(g_H(v), x) = d(v, x)$. Gated sets are convex by definition.

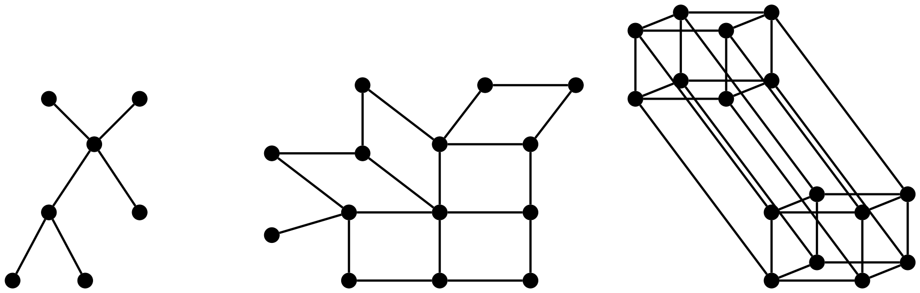
Given an integer $k \geq 1$, the hypercube of dimension k, Q_k, is a graph representing all the subsets of $\{1, \ldots, k\}$ as the vertex set. An edge connects two subsets if one is included into the other and they differ by only one element. Hypercube Q_2 is a *square* and Q_3 is a 3-cube.

▶ **Definition 2.1** (Median graph). *A graph is median if, for any triplet x, y, z of distinct vertices, the set $I(x, y) \cap I(y, z) \cap I(z, x)$ contains exactly one vertex $m(x, y, z)$ called the* median *of x, y, z.*

Observe that certain well-known families of graphs are median: trees, grids, squaregraphs [7], and hypercubes Q_k. Median graphs are bipartite and do not contain an induced $K_{2,3}$ [5, 26, 34]. They can be obtained by Mulder's convex expansion [34, 35] starting from a single vertex.

Now, we define a parameter which has a strong influence on the study of median graphs. The dimension $d = \dim(G)$ of a median graph G is the dimension of the largest hypercube contained in G as an induced subgraph. In other words, G admits Q_d as an induced subgraph, but not Q_{d+1}. Median graphs with $d = 1$ are exactly the trees. Median graphs with $d \leq 2$ are called *cube-free* median graphs.

Figure 1 presents three examples of median graphs. (a) is a tree: $d = 1$. (b) is a cube-free median graph: it has dimension $d = 2$. To be more precise, it is a squaregraph [7], which is a sub-family of cube-free median graphs. The last one (c) is a 4-cube: it has dimension $d = 4$.



(a) Tree, $d = 1$. **(b)** Squaregraph, $d = 2$. **(c)** 4-cube, $d = 4$.

Figure 1 Examples of median graphs.

We provide a list of properties satisfied by median graphs. In particular, we define the notion of Θ-classes which is a key ingredient of several existing algorithms [13, 25, 27].

In general graphs, all gated subgraphs are convex. The reverse is true in median graphs.

▶ **Lemma 2.2** (Convex⇔Gated [5, 13])**.** *Any convex subgraph of a median graph is gated.*

To improve readibility, edges $(u, v) \in E$ are sometimes denoted by uv. We remind the notion of Θ-class, which is well explained in [13], and enumerate some properties related to it. We say that the edges uv and xy are in relation Θ_0 if they form a square $uvyx$, where uv and xy are opposite edges. Then, Θ refers to the reflexive and transitive closure of relation Θ_0. Let q be the number of equivalence classes obtained with this relation. The classes of the equivalence relation Θ are denoted by $E_1, \ldots, E_q$. Concretely, two edges uv and $u'v'$ belong to the same Θ-class if there is a sequence $uv = u_0v_0, u_1v_1, \ldots, u_rv_r = u'v'$ such that u_iv_i and $u_{i+1}v_{i+1}$ are opposite edges of a square. We denote by $\mathcal{E}$ the set of Θ-classes: $\mathcal{E} = \{E_1, \ldots, E_q\}$. To avoid confusions, let us highlight that parameter q is different from the dimension d: for example, on trees, $d = 1$ whereas $q = n - 1$. Moreover, the dimension d is at most $\lfloor \log n \rfloor$ in general.

▶ **Lemma 2.3** (Θ-classes in linear time [13])**.** *There exists an algorithm which computes the* Θ*-classes* $E_1, \ldots, E_q$ *of a median graph in linear time* $O(|E|) = O(dn)$*.*

In median graphs, each class E_i, $1 \leq i \leq q$, is a perfect matching cutset and its two sides H_i' and H_i'' verify nice properties, that are presented below.

▶ **Lemma 2.4** (Halfspaces of E_i [13, 25, 35])**.** *For any* $1 \leq i \leq q$*, the graph* G *deprived of edges of* E_i*,* i.e. $G \backslash E_i = (V, E \backslash E_i)$*, has two connected components* H_i' *and* H_i''*, called halfspaces. Edges of* E_i *form a matching: they have no endpoint in common. Halfspaces satisfy the following properties.*

- *Both* H_i' *and* H_i'' *are convex/gated.*
- *If* uv *is an edge of* E_i *with* $u \in H_i'$ *and* $v \in H_i''$*, then* $H_i' = W(u, v) = \{x \in V : d(x, u) < d(x, v)\}$ *and* $H_i'' = W(v, u) = \{x \in V : d(x, v) < d(x, u)\}$*.*

We denote by $\partial H_i'$ the subset of H_i' containing the vertices which are adjacent to a vertex in H_i'': $\partial H_i' = N(H_i'')$. Put differently, set $\partial H_i'$ is made up of vertices of H_i' which are endpoints of edges in E_i. Symmetrically, set $\partial H_i''$ contains the vertices of H_i'' which are adjacent to H_i'. We say these sets are the *boundaries* of halfspaces H_i' and H_i'' respectively. Figure 2 illustrates the notions of Θ-class, halfspace and boundary on a small example. In this particular case, an halfspace is equal to its boundary: $\partial H_i'' = H_i''$. The vertices of $\partial H_i'$ are colored in blue.

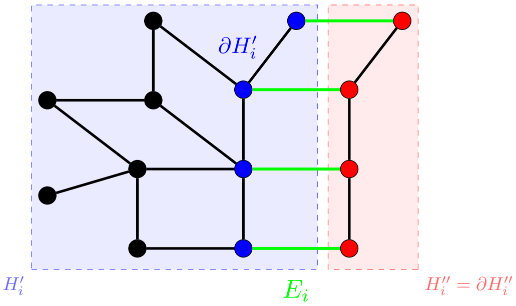


Figure 2 A class E_i with sets $H_i', H_i'', \partial H_i', \partial H_i''$.

▶ **Lemma 2.5** (Boundaries [13, 25, 35]). *Both $\partial H_i'$ and $\partial H_i''$ are convex/gated. Moreover, the edges of E_i define an isomorphism between $\partial H_i'$ and $\partial H_i''$.*

As a consequence, suppose uv and $u'v'$ belong to E_i: if uu' is an edge and belongs to class E_j, then vv' is an edge too and it belongs to E_j. We terminate this list of lemmas with a last property dealing with the orientation of edges from a canonical basepoint $v_0 \in V$. The v_0-*orientation* of the edges of G according to v_0 is such that, for any edge uv, the orientation is $\overrightarrow{uv}$ if $d(v_0, u) < d(v_0, v)$. Indeed, we cannot have $d(v_0, u) = d(v_0, v)$ as G is bipartite. The v_0-orientation is acyclic.

▶ **Lemma 2.6** (Orientation [13]). *All edges can be oriented according to any canonical basepoint v_0.*

From now on, we suppose that an arbitrary basepoint $v_0 \in V$ has been selected and we refer automatically to the v_0-orientation when we mention incoming or outgoing edges.

2.2 Shortest paths and signature

We fix an arbitrary canonical basepoint v_0 and for each class E_i, we say that the halfspace containing v_0 is H_i'. Given two vertices $u, v \in V$, we define the set which contains the Θ-classes separating u from v.

▶ **Definition 2.7** (Signature $\sigma_{u,v}$). *We say that the* signature *of the pair of vertices u, v, denoted by $\sigma_{u,v}$, is the set of classes E_i such that u and v are separated in $G \backslash E_i$. In other words, u and v are in different halfspaces of E_i.*

The signature of two vertices provide us with the composition of any shortest (u, v)-path. Indeed, all shortest (u, v)-paths contain exactly one edge for each class in $\sigma_{u,v}$.

▶ **Lemma 2.8** ([15]). *For any shortest (u, v)-path P, the edges in P belong to classes in $\sigma_{u,v}$ and, for any $E_i \in \sigma_{u,v}$, there is exactly one edge of E_i in path P. Conversely, a path containing at most one edge of each Θ-class is a shortest path between its departure and its arrival.*

This result is a direct consequence of the convexity of halfspaces. By contradiction, a shortest path that would pass through two edges of some Θ-class E_i would escape temporarily from an halfspace, say w.l.o.g H_i', which is convex (Lemma 2.4).

Definition 2.7 can be generalized: given a set of edges $B \subseteq E$, its signature is the set of Θ-classes represented in that set: $\{E_i : uv \in E_i \cap B\}$. The signature of a path is the set of classes which have at least one edge in this path. In this way, the signature $\sigma_{u,v}$ is also the signature of any shortest (u,v)-path. The signature of a hypercube is the set of Θ-classes represented in its edges: the cardinality of the signature is thus equal to the dimension of the hypercube.

2.3 Orthogonal Θ-classes and hypercubes

We present now another important notion on median graphs: *orthogonality.*

▶ **Definition 2.9** (Orthogonal Θ-classes). *We say that classes E_i and E_j are* orthogonal *(denoted by $E_i \perp E_j$) if there is a square $uvyx$ in G, where $uv, xy \in E_i$ and $ux, vy \in E_j$.*

We say that E_i and E_j are *parallel* if they are not orthogonal, that is $H_i \subseteq H_j$ for some $H_i \in \{H_i', H_i''\}$, $H_j \in \left\{H_j', H_j''\right\}$. We define the sets of pairwise orthogonal Θ-classes.

▶ **Definition 2.10** (Pairwise Orthogonal Family). *We say that a set of classes $X \subseteq \mathcal{E}$ is a* Pairwise Orthogonal Family (POF for short) *if for any pair $E_j, E_h \in X$, we have $E_j \perp E_h$.*

For any induced hypercube of G, its *basis* (resp. *anti-basis*) is the closest vertex (resp. farthest) to v_0 in it. All edges of the hypercube indicent to the basis are outgoing from it in the v_0-orientation. Hypercubes are in bijection with pairs (u, L), where u is a vertex (the basis of the hypercube) and L is a POF outgoing from u (the *signature* of the hypercube).

The full version of this subsection is put in Appendix A.

3 Simplex graphs

Due to page limit, the proofs in this Section are omitted. Given any undirected graph G, the vertices of the simplex graph $K(G)$ associated to G represent the induced cliques (not necessarily maximal) of G. Two of these cliques are connected by an edge if they differ by exactly one element.

▶ **Definition 3.1** (Simplex graphs [9]). *The simplex graph $K(G) = (V_K, E_K)$ of $G = (V, E)$ is made up of the vertex set $V_K = \{C \subseteq V : C$ induced complete graph of $G\}$ and the edge set $E_K = \{(C, C') : C, C' \in V_K, C \subsetneq C', |C'| - |C| = 1\}$.*

Simplex graphs can be characterized as particular median graphs.

▶ **Theorem 3.2.** *Let G be a median graph. The following statements are equivalent:*

(1) *G is a simplex graph.*

(2) *There is a vertex $v_0 \in V(G)$ such that each Θ-class of G is adjacent to v_0,* i.e. $\forall 1 \leq i \leq q, \exists v_i \in V(G), v_0v_i \in E_i$.

(3) *There is a vertex $v_0 \in V(G)$ contained in any maximal hypercube of G.*

In this section only, on simplex graphs, the canonical basepoint v_0 is not selected arbitrarily. We fix v_0 as a vertex adjacent to all Θ-classes, as put in evidence by Theorem 3.2. We call v_0 the *central vertex* of the simplex graph.

▶ **Definition 3.3** (Crossing graphs [9, 28]). *Let G be a median graph. Its crossing graph $G^\#$ is the graph obtained by considering Θ-classes as its vertices and such that two Θ-classes are adjacent if they are orthogonal.*

Restricted to simplex graphs, this transformation is the reverse of K: indeed, as stated in [28], $G = K(G)^\#$. The clique number of $G^\#$ is exactly the dimension of median graph G. For example, the crossing graph of a cube-free median graph contains no triangle. Each simplex graph admits a central vertex (v_0 in Theorem 3.2) which represents the empty clique of $G^\#$.

Now, we focus on the problem of determining a diametral pair of a simplex graph G and more generally all eccentricities. Observe that the distance between the central vertex v_0 and any vertex u of G can be deduced directly from the set $\mathcal{E}^-(u)$ of Θ-classes incoming into u. We state that $\sigma_{v_0,u} = \mathcal{E}^-(u)$. This is a consequence of Theorem 3.2: all Θ-classes of $\mathcal{E}^-(u)$ are adjacent to v_0, so v_0 is the basis of the hypercube with signature $\mathcal{E}^-(u)$ and anti-basis u. A shortest (v_0, u)-path is thus made up of edges of this hypercube. The distance $d(v_0, u)$ is equal to its dimension: $d(v_0, u) = |\mathcal{E}^-(u)|$.

A key result is the fact that the central vertex v_0 of the simplex graph belongs to the interval $I(u, v)$ of any pair u, v satisfying $d(u, v) = \mathsf{ecc}(u)$.

▶ **Lemma 3.4.** *Let $u, v \in V(G)$ such that $d(u, v) = \mathsf{ecc}(u)$. Then, $v_0 \in I(u, v)$.*

Two vertices u, v forming a diametral pair cannot share a common incoming Θ-class E_i, in other words $\mathcal{E}^-(u) \cap \mathcal{E}^-(v) = \emptyset$, otherwise $m = m(u, v, v_0) \in I(u, v) \subseteq H_i''$ and $v_0 \in H_i'$. Moreover, the distance $d(u, v)$ is exactly $|\mathcal{E}^-(u)|+|\mathcal{E}^-(v)|$ because $|\mathcal{E}^-(u)| = d(v_0, u)$ and $|\mathcal{E}^-(v)| = d(v_0, v)$. So, determining the diameter of a simplex graph G is equivalent to maximizing the sum $|X| + |Y|$, where X and Y are two POFs of G that are disjoint. Computing the diameter is equivalent to find the largest pair of disjoint cliques in the crossing graph $G^{\#}$. Similarly, the eccentricity of a vertex u is exactly the size $|\mathcal{E}^-(u)| + |\mathcal{E}^-(v)|$ of the largest pair of disjoint POFs $(\mathcal{E}^-(u), \mathcal{E}^-(v))$. Now, we can define the notion of *opposite*.

▶ **Definition 3.5.** *Let G be a simplex graph and X a POF of G. We denote by $\mathsf{op}(X)$ the opposite of X,* i.e. *the POF Y disjoint from X with the maximum cardinality.*

$$\mathsf{op}(X) = \operatorname*{argmax}_{Y \cap X = \emptyset} |Y| \,.$$

With this definition, the eccentricity of a vertex u, if we fix $X_u = \mathcal{E}^-(u)$, is written $\mathsf{ecc}(u) = |X_u| + |\mathsf{op}(X_u)|$. Hence, the diameter of the simplex graph G can be written as the size of the largest pair POF-opposite: $\mathsf{diam}(G) = \max_{X \in \mathcal{L}}(|X| + |\mathsf{op}(X)|)$.

We propose now the definition of two problems on simplex graphs. The first one, called Opposites (OPP) consists in finding all pairs POF-opposite. Its output has thus a linear size. Given the solution of OPP on graph G, one can deduce both the diameter and all eccentricities in $O(n)$ time with the formula: $\mathsf{ecc}(u) = |X_u| + |\mathsf{op}(X_u)|$.

▶ **Definition 3.6** (OPP).
Input: *Simplex graph G, central vertex v_0.*
Output: *For each POF X, its opposite $\mathsf{op}(X)$.*

We define an even larger version of the problem where a positive integer weight is associated with each POF. We call it Weighted Opposites (WOPP).

▶ **Definition 3.7** (WOPP).
Input: *Simplex graph G, central vertex v_0, weight function $\omega : \mathcal{L} \to \mathbb{N}^+$.*
Output: *For each POF X, its weighted opposite Y maximizing $\omega(Y)$ such that $X \cap Y = \emptyset$.*

Obviously, OPP is a special case of WOPP when ω is the cardinality function. We show that WOPP can be solved in quasilinear time $O((d^3 + \log n)n)$, as $d \leq \lfloor \log n \rfloor$. As a consequence, all eccentricities of a simplex graph G can also be determined with such time complexity.

▶ **Theorem 3.8.** *There is a combinatorial algorithm solving WOPP in time $O((d^3 + \log n)n)$. Consequently, it determines all eccentricities of a simplex graph with this running time.*

4 Subquadratic-time algorithm for all eccentricities on median graphs

This section introduces the design of algorithms computing all eccentricities for the whole class of median graphs (not only simplex graphs). We begin in Section 4.1 with the proposal of a linear-time FPT algorithm, parameterized by the dimension d, running in $2^{O(d)}n$. It is based on some techniques of a paper of the literature [15] which provides a slightly super-exponential time algorithm - running in $2^{O(d \log d)}n$ - for the same problem. We prove that replacing one step of this procedure by the partitioning conceived in Section 3 allows us to decrease the exponential dependence on d.

Thanks to this outcome, in Section 4.2, we are able to design a first subquadratic-time algorithm for all median graphs running in $\tilde{O}(n^{\frac{5}{3}})$. The proof of the Lemmas which are not given in this subsection are put in Appendix B.

4.1 Linear FPT algorithm for constant-dimension median graphs

We remind in this subsection the different steps needed to obtain a linear-time algorithm computing all eccentricities of a median graph with constant dimension, $d = O(1)$. We show how Theorem 3.8 can be integrated to it in order to improve the dependence on d. Let us begin with a reminder of the former result.

▶ **Lemma 4.1** ([15]). *There is a combinatorial algorithm computing all eccentricities in a median graph G with running time $\tilde{O}(2^{d(\log d+1)}n)$.*

Some parts of this subsection are redundant with [15], however we keep this subsection self-contained. The new outcomes presented are Theorems 4.10 and 4.16.

The algorithm evoked by Lemma 4.1 consists in the computation of three kinds of labels: *ladder* labels φ, *opposite* labels op and *anti-ladder* labels ψ. The order in which they are given correspond to their respective dependence: op-labelings are functions of labels φ and ψ-labelings are functions of both labels φ and op. The definition of op-labelings on general median graphs is closely related to the computation of eccentricities on simplex graphs evoked in Section 3.

4.1.1 Ladder labels

Some preliminary work has to be done before giving the definition of labels φ. We introduce the notion of *ladder set*. It is defined only for pairs of vertices u, v satisfying the condition $u \in I(v_0, v)$. In this situation, the edges of shortest (u, v)-paths are all oriented towards v with the v_0-orientation.

▶ **Definition 4.2** (Ladder set $L_{u,v}$). *Let $u, v \in V$ such that $u \in I(v_0, v)$. The ladder set $L_{u,v}$ is the subset of $\sigma_{u,v}$ which contains the Θ-classes admitting an edge adjacent to u.*

Figure 3 shows a small median graph with four vertices v_0, u, v, x such that $u \in I(v_0, v)$ and $u \in I(v_0, x)$. It gives the composition of ladder sets $L_{u,v}$ and $L_{u,x}$.

A key characterization on ladder sets states that their Θ-classes are pairwise orthogonal. In brief, every set $L_{u,v}$ is a POF. Let us remind that the adjacency of all Θ-classes of a POF L with the same vertex u implies the existence of a (unique) hypercube not only signed with this POF L but also containing u (Lemma A.5). If additionnally POF L is *outgoing from* u - said differently, the edges adjacent to u belonging to a Θ-class of L leave u -, then u is the basis of the hypercube. As the Θ-classes of $L_{u,v}$ are adjacent to u by definition, there is a natural bijection between (i) hypercubes (ii) pairs made up of a vertex u and a POF L outgoing from u and (iii) pairs vertex-ladder set $(u, L_{u,\cdot})$.

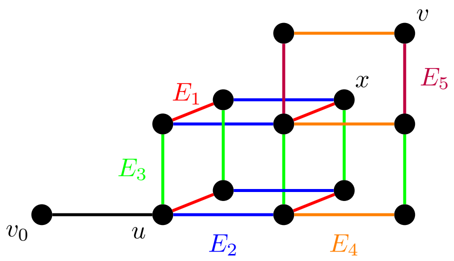


Figure 3 Examples of ladder sets: $L_{u,v} = \{E_2, E_3\}$, $L_{u,x} = \{E_1, E_2, E_3\}$.

▶ **Lemma 4.3** ([15]). *Every ladder set $L_{u,v}$ is a POF. For any ordering τ of the Θ-classes in $L_{u,v}$, there is a shortest (u, v)-path such that, for any $1 \le i \le |L_{u,v}|$, the i^{th} first edge of the path belongs to the i^{th} Θ-class of $L_{u,v}$ in ordering τ.*

The necessary background to introduce labels φ is now known.

▶ **Definition 4.4** (Labels φ [15]). *Given a vertex u and a POF L outgoing from u, let $\varphi(u, L)$ be the maximum distance $d(u, v)$ such that $u \in I(v_0, v)$ and $L_{u,v} = L$.*

Intuitively, integer $\varphi(u, L)$ provides us with the maximum length of a shortest path starting from u into "direction" L. Observe that the total size of labels φ on a median graph G does not exceed $O(2^d n)$, according to Lemma A.7. We provide another notion related to orthogonality which will be used in the remainder.

▶ **Definition 4.5** (L-parallelism). *We say that a POF L' is L-*parallel *if, for any $E_j \in L'$, $L \cup \{E_j\}$ is not a POF.*

When L' is a L-parallel POF, we have $L \cap L' = \emptyset$, otherwise $L \cup \{E_j\} = L$ for some $E_j \in L'$. Presented differently, a L-parallel POF is such that any of its Θ-classes is parallel to at least one Θ-class of L.

A combinatorial algorithm running in $\tilde{O}(2^{2d} n)$ which computes all labels $\varphi(u, L)$ was identified in [15]: we provide an overview of it. There is a crucial relationship between a label $\varphi(u, L)$ and the labels of (i) the anti-basis u^+ of the hypercube with basis u and signature L and (ii) the L-parallel POFs outgoing from u^+.

▶ **Lemma 4.6** (Inductive formula for labels φ [15]). *Let $u \in V$, L be a POF outgoing from u and Q be the hypercube with basis u and signature L. We denote by u^+ the opposite vertex of u in Q: u is the basis of Q and u^+ its anti-basis. A vertex $v \neq u^+$ verifies $u \in I(v_0, v)$ and $L_{u,v} = L$ if and only if (i) $u^+ \in I(v_0, v)$ and (ii) ladder set $L_{u^+,v}$ is L-parallel.*

A consequence of the previous lemma is that we can distinguish two cases for the computation of $\varphi(u, L)$. In the first case, $\varphi(u, L) = |L|$: it occurs when the farthest-to-u vertex with ladder set L is u^+ (base case). Indeed, u^+ is a candidate as $\sigma_{u,u^+} = L$: shortest (u, u^+)-paths pass through hypercube Q. This situation happens when either no Θ-class is outgoing from u^+ or when all Θ-classes outgoing from u^+ are orthogonal to L. In the second case, there are vertices farther to u than u^+ with ladder set L. As announced in Lemma 4.6, $\varphi(u, L)$ is a function of labels $\varphi(u^+, \cdot)$.

$$\varphi(u, L) = \max_{\substack{L^+ \text{ POF outgoing from } u^+ \\ \forall E_j \in L^+,\ L \cup \{E_j\} \text{ not POF}}} (|L| + \varphi(u^+, L^+)). \tag{1}$$

If there exists such a POF L^+, then the label $\varphi(u, L)$ is given by Equation (1). Otherwise, it is given by the first case. Briefly, the algorithm consists in listing all pairs vertex-ladder set $((u, L), (u^+, L^+))$ such that u^+ is the anti-basis of the hypercube of basis u and signature L. For each of it, we verify whether L^+ is L-parallel. If it is, we update $\varphi(u, L)$ if $|L| + \varphi(u^+, L^+)$ is greater than the current value. The total number of pairs $((u, L), (u^+, L^+))$ is upper-bounded by $2^{2d}n$: there are at most $2^d n$ pairs (u^+, L^+) (bijection with hypercubes) and, for each of them, there are at most 2^d compatible pair (u, L) such that u^+ is the anti-basis of (u, L). Indeed, the number of edges incoming into u^+ is at most d (Lemma A.6). For this reason, the computation of φ-labelings takes $\tilde{O}(2^{2d}n)$.

▶ **Theorem 4.7** (Computation of labels φ [15]). *There is a combinatorial algorithm which determines all labels $\varphi(u, L)$ in $\tilde{O}(2^{2d}n)$. It also stores, for each pair (u, L), a vertex v satisfying $L_{u,v} = L$ and $d(u, v) = \varphi(u, L)$, denoted by $\mu(u, L)$.*

4.1.2 Opposite labels

The second type of labels needed to compute all eccentricities of a median graph G are opposite labels. Given a vertex u and a POF L outgoing from u, let $\mathsf{op}_u(L)$ denote a POF outgoing from u with maximum label φ which is disjoint from L. As for φ, the total size of op-labelings is $O(2^d n)$.

▶ **Definition 4.8** (Labels op [15]). *Let $u \in V$ and L be a POF outgoing from u. Let $\mathsf{op}_u(L)$ be one of the POF L' outgoing from u, disjoint from L, which maximizes value $\varphi(u, L')$.*

On simplex graphs, the opposite function provides in fact the op-labelings of vertex v_0: $\mathsf{op}(X) = \mathsf{op}_{v_0}(X)$. As all vertices belong to hypercubes with basis v_0, the ladder set $L_{v_0,v}$ for any vertex $v \in V$ is exactly the set $\mathcal{E}^-(v)$ of Θ-classes incoming into v. So, value $\varphi(v_0, X)$ is the distance $d(v_0, v)$ between v_0 and the only vertex v with ladder set $L_{v_0,v} = X$.

On general median graphs, the opposite label $\mathsf{op}_u(L)$ allows us to obtain the maximum distance $d(s, t)$ such that $u = m(s, t, v_0)$ and the ladder set $L_{u,s}$ is L.

▶ **Lemma 4.9** (Relationship between medians and disjoint outgoing POFs [15]). *Let L, L' be two POFs outgoing from a vertex u. Let s (resp. t) be a vertex such that $u \in I(v_0, s)$ (resp. $u \in I(v_0, t)$) and $L_{u,s} = L$ (resp. $L_{u,t} = L'$). Then, $u \in I(s, t)$ if and only if $L \cap L' = \emptyset$. Therefore, given a single vertex s such that $u \in I(v_0, s)$ and $L_{u,s} = L$, the maximum distance $d(s, v)$ we can have with median $m(s, v, v_0) = u$ is exactly $d(u, s) + \varphi(u, \mathsf{op}_u(L))$.*

Going further, given a vertex $u \in V$, the maximum distance $d(s, t)$ such that $u = m(s, t, v_0)$ is the maximum value $\varphi(u, L) + \varphi(u, \mathsf{op}_u(L))$, where L is a POF outgoing from u.

An algorithm was initially proposed to compute all labels $\mathsf{op}_u(L)$ consisting in a brute force bounded tree search [15]. Its execution time was $\tilde{O}(2^{O(d \log d)}n)$, leading to the global same asymptotic running time (Lemma 4.1) for finding all eccentricities.

Fortunately, the quasilinear time algorithm determining the eccentricities on simplex graphs (Theorem 3.8, Section 3) offers us the opportunity to decrease the exponential term to a simple exponential function 2^d. For any $u \in V$, let $G_u = G[V_u]$ be the *star* graph of u, using a definition from [22]. Its vertex set V_u is made up of the vertices belonging to a hypercube with basis u in G. Graph G_u is the induced subgraph of G on vertex set V_u (see Figure 8 for an example). Chepoi *et al.* [22] showed that graph G_u is a gated/convex subgraph of G. Applying the algorithm of Theorem 3.8 on the simplex graph G_u provides us with the opposite labels of u.

▶ **Theorem 4.10** (B.1, Computation of labels op). *There is a combinatorial algorithm which determines all labels $\mathsf{op}_u(L)$ in $\tilde{O}(2^d n)$.*

4.1.3 Anti-ladder labels

We terminate with anti-ladder labels ψ which play the converse role of ladder labels φ. While $\varphi(u, L)$ is defined for POFs L outgoing from u, labels $\psi(u, R)$ are defined for POFs R incoming into u, *i.e.* every Θ-class of the POF has an edge entering u. As any such pair (u, R) can be associated with a hypercube of anti-basis u and signature R (Lemma A.6), the total size of ψ-labelings is at most $O(2^d n)$ too.

The notion of *milestone* intervenes in the definition of labels ψ. We consider two vertices u, v such that $u \in I(v_0, v)$. Milestones are defined recursively.

▶ **Definition 4.11** (Milestones $\Pi(u, v)$)**.** *Let $L_{u,v}$ be the ladder set of u, v and u^+ be the anti-basis of the hypercube with basis u and signature $L_{u,v}$. If $u^+ = v$, then pair u, v admits two milestones: $\Pi(u, v) = \{u, v\}$. Otherwise, the set $\Pi(u, v)$ is the union of $\Pi(u^+, v)$ with vertex u: $\Pi(u, v) = \{u\} \cup \Pi(u^+, v)$.*

The milestones are the successive anti-bases of the hypercubes formed by the vertices and ladder sets traversed from u to v. Both vertices u and v are contained in $\Pi(u, v)$. The first milestone is u, the second is the anti-basis u^+ of the hypercube with basis u and signature $L_{u,v}$. The third one is the anti-basis u^{++} of the hypercube with basis u^+ and signature $L_{u^+,v}$, etc. All milestones are metrically between u and v: $\Pi(u, v) \subseteq I(u, v)$.

▶ **Definition 4.12** (Penultimate milestone $\overline{\pi}(u, v)$)**.** *We say that the milestone in $\Pi(u, v)$ different from v but the closest to it is called the penultimate milestone. We denote it by $\overline{\pi}(u, v)$. Furthermore, we denote by $\overline{L}_{u,v}$ the* anti-ladder set *of u, v,* i.e. *the Θ-classes of the hypercube with basis $\overline{\pi}(u, v)$ and anti-basis v.*

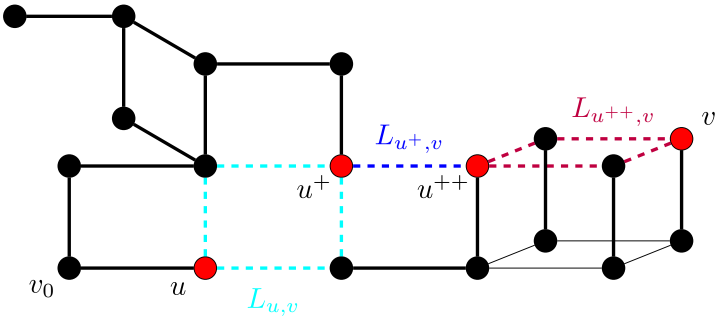


Figure 4 A pair u, v with $u \in I(v_0, v)$ and its milestones $\Pi(u, v)$ in red.

Figure 4 shows the milestones $\Pi(u, v) = \{u, u^+, u^{++}, v\}$. The hypercubes with the following pair basis-signature are highlighted with dashed edges: $(u, L_{u,v})$, $(u^+, L_{u^+,v})$, and $(u^{++}, L_{u^{++},v})$. We have $\overline{\pi}(u, v) = u^{++}$ and $\overline{L}_{u,v} = L_{u^{++},v}$ is drawn in purple.

Let R be a POF incoming to some vertex u and u^- be the basis of the hypercube with anti-basis u and signature R. Label $\psi(u, R)$ intuitively represents the maximum distance of a shortest path arriving to vertex u from "direction" R.

▶ **Definition 4.13** (Labels ψ [15])**.** *The label $\psi(u, R)$ is the maximum distance $d(u, v)$ we can obtain with a vertex v satisfying the following properties:*

- *$m = m(u, v, v_0) \neq u$,*
- *the anti-ladder set of m, u is R: $\overline{L}_{m,u} = R$.*

Equivalently, vertex u^- is the penultimate milestone of pair m, u: $u^- = \overline{\pi}(m, u)$.

As for the computation of labels φ, there is an induction process to determine all $\psi(u, R)$. As the base case, suppose that $u^- = v_0$. The largest distance $d(u, v)$ we can obtain with a vertex v such that $v_0 \in I(u,v)$ consists in considering the opposite $\mathsf{op}_{v_0}(R)$ of R which is outgoing from v_0. Hence, $\psi(u,R) = |R| + \varphi(v_0, \mathsf{op}_{v_0}(R))$.

For the induction step, we distinguish two cases. In the first one, assume that $m(u,v,v_0) = u^-$ - equivalently, $\Pi(m,u) = \Pi(u^-,u) = \{u^-,u\}$. A shortest (u,v)-path is the concatenation of the shortest (u,u^-)-path of length $|R|$ with a shortest (u^-,v)-path, and $u^- \in I(v_0,v)$. The largest distance $d(u,v)$ we can have, as for the base case, is $\psi(u,R) = |R| + \varphi(u^-, \mathsf{op}_{u^-}(R))$.

In the second case, $m \neq u^-$, an inductive formula allows us to obtain $\psi(u,R)$. A consequence of Lemma 4.6 is that, for two consecutive milestones in $\Pi(u,v)$, say u and u^+ w.l.o.g, then $L_{u^+,v}$ is $L_{u,v}$-parallel. This observation, applied to the penultimate milestone, provides us with the following theorem.

▶ **Lemma 4.14** (Inductive formula for labels ψ [15]). *Let $u, v \in V$ and $u \in I(v_0,v)$. Let L be a POF outgoing from v and w the anti-basis of hypercube (v,L). The following propositions are equivalent:*

(i) *vertex v is the penultimate milestone of (u,w): $\overline{\pi}(u,w) = v$,*
(ii) *the milestones of (u,w) are the milestones of (u,v) with w: $\Pi(u,w) = \Pi(u,v) \cup \{w\}$,*
(iii) *the POF L is $\overline{L}_{u,v}$-parallel.*

Set $\Pi(m,u)$ admits at least three milestones: m, u^-, and u. Let R^- be the POF incoming to u^- which is the ladder set (but also the signature) of (i) the milestone just before u^- and (ii) u^-. According to Lemma 4.14, vertex u^- is the penultimate milestone of (m,u) if and only if $R^- \cup \{E_j\}$ is not a POF, for each $E_j \in R$. For this reason, value $\psi(u,R)$ can be expressed as:

$$\psi(u,R) = \max_{\substack{R^- \text{ POF incoming to } u^- \\ \forall E_j \in R, R^- \cup \{E_j\} \text{ not POF}}} (|R| + \psi(u^-,R^-)) \tag{2}$$

Our algorithm consists in taking the maximum value between the two cases. The number of pairs $((u,R),(u^-,R^-))$ which satisfy the condition described in Equation (2) is at most $2^{2d}n$: it is identical to the one presented for φ-labelings.

▶ **Theorem 4.15** (Computation of labels ψ [15]). *There is a combinatorial algorithm which determines all labels $\psi(u,R)$ in $\tilde{O}(2^{2d}n)$.*

4.1.4 Better time complexity for all eccentricities

The computation of all labels $\varphi(u,L)$, $\mathsf{op}_u(L)$ and $\psi(u,R)$ gives an algorithm which determines all eccentricities. Indeed, each eccentricity $\mathsf{ecc}(u)$ is a function of certain labels φ and ψ. Let v be a vertex in G such that $\mathsf{ecc}(u) = d(u,v)$. If $m = m(u,v,v_0) = u$, then $u \in I(v_0,v)$ and value $d(u,v)$ is given by a label $\varphi(u,L)$. Otherwise, if $m \neq u$, let u^- be the penultimate milestone in $\Pi(m,u)$ and R be the classes of the hypercube with basis u^- and anti-basis u. The eccentricity of u is given by a label $\psi(u,R)$. Conversely, each $\varphi(u,L)$ and $\psi(u,R)$ is the distance between u and another vertex by definition. Therefore, we have:

$$\mathsf{ecc}(u) = \max \left\{ \max_{\substack{L \text{ POF} \\ \text{outgoing from } u}} \varphi(u,L), \max_{\substack{R \text{ POF} \\ \text{incoming to } u}} \psi(u,R) \right\} \tag{3}$$

In other words, the eccentricity of u is the maximum label φ or ψ centered at u. We can conclude with the main result of this subsection: the eccentricities of any median graph can be determined in linear time multiplied by a simple exponential function $2^{O(d)}$ of the dimension d.

▶ **Theorem 4.16** (All eccentricities in $\tilde{O}(2^{2d}n)$-time for median graphs). *There is a combinatorial algorithm computing the list of all eccentricities of a median graph G in time $\tilde{O}(2^{2d}n)$.*

4.2 Tackling the general case

Our new FPT algorithm for computing the list of eccentricities in a median graph has a runtime in $2^{\mathcal{O}(d)}n$, with d being the dimension (Theorem 4.16). This runtime stays subquadratic in n as long as $d < \alpha \log n$, for some constant $\alpha < 1$. In what follows, we present a simple partitioning scheme for median graphs into convex subgraphs of dimension at most $\alpha \log n$, for an arbitrary value of $\alpha \leq 1$. By doing so, we obtain (in combination with Theorem 4.16) the first known subquadratic-time algorithm for computing all eccentricities in a median graph.

We start with a simple relation between the eccentricity function of a median graph and the respective eccentricity functions of any two complementary halfspaces.

▶ **Lemma 4.17** (B.2). *Let G be a median graph. For every $1 \leq i \leq q$, let $v \in V(H_i')$ be arbitrary, and let $v^* \in \partial H_i''$ be its gate. Then, $\mathsf{ecc}(v) = \max\{\mathsf{ecc}_{H_i'}(v), d(v, v^*) + \mathsf{ecc}_{H_i''}(v^*)\}$.*

We will use this above Lemma 4.17 later in our proof in order to compute in linear time the list of eccentricities in a median graph being given the lists of eccentricities in any two complementary halfspaces.

Next, we give simple properties of Θ-classes, to be used in the analysis of our main algorithm in this section (see Lemma 4.21).

▶ **Lemma 4.18** (B.3). *Let H and G be median graphs. If H is an induced subgraph of G then, every Θ-class of H is contained in a Θ-class of G.*

This above Lemma 4.18 can be strenghtened in the special case of *isometric* subgraphs.

▶ **Lemma 4.19** (B.4). *Let H and G be median graphs, and let $E_1, E_2, \ldots, E_q$ denote the Θ-classes of G. If H is an isometric subgraph of G then, the Θ-classes of H are exactly the nonempty subsets among $E_i \cap E(H)$, for $1 \leq i \leq q$.*

An important consequence of Lemma 4.18 is the following relation between the dimension d of a median graph and the cardinality of its Θ-classes.

▶ **Lemma 4.20.** *Let G be a median graph, and let $D := \max\{|E_i| \mid 1 \leq i \leq q\}$ be the maximum cardinality of a Θ-class of G. Then, $d = \dim(G) \leq \lfloor \log D \rfloor + 1$.*

Proof. Any induced d-dimensional hypercube of G contains exactly 2^{d-1} edges of its Θ-classes, so $2^{d-1} \leq D$. ◀

We are now ready to present our main technical contribution in this section.

▶ **Lemma 4.21.** *If there is an algorithm for computing all eccentricities in an n-vertex median graph of dimension at most d in $\tilde{O}(c^d \cdot n)$ time, then in $\tilde{O}(n^{2-\frac{1}{1+\log c}})$ time we can compute all eccentricities in* any *n-vertex median graph.*

Proof. Let G be an n-vertex median graph. We compute its Θ-classes $E_1, E_2, \ldots, E_q$, that takes linear time (Lemma 2.3). For some parameter D (to be fixed later in the proof), let us assume without loss of generality $E_1, E_2, \ldots, E_p$ to be the subset of all Θ-classes of cardinality $\geq D$, for some $p \leq q$. Note that we have $p \leq |E| / D = \tilde{O}(n/D)$.

We reduce the problem of computing all eccentricities in G to the same problem on every connected component of $G \setminus (E_1 \cup E_2 \cup \ldots \cup E_p)$. More formally, we construct a rooted binary tree T, whose leaves are labelled with convex subgraphs of G. Initially, T is reduced to a single node with label equal to G. Then, for $i = 1 \ldots p$, we further refine this tree so that, at the end of any step i, its leaves are labelled with the connected components of $G \setminus (E_1 \cup E_2 \cup \ldots \cup E_i)$. An example of T is shown in Figure 5 with $D = 3$ and two Θ-classes reaching this cardinality bound.

For that, we proceed as follows. We consider all leaves of T whose label H satisfies $E(H) \cap E_i \neq \emptyset$. By Lemma 4.19, $E(H) \cap E_i$ is a Θ-class of H. Both halfspaces of E_i become the left and right children of H in T. Recall that the leaves of T at this step i are the connected components of $G \setminus (E_1 \cup E_2 \cup \ldots \cup E_{i-1})$, and in particular that they form a partition of $V(G)$. Therefore, each step takes linear time by reduction to computing the connected components in vertex-disjoint subgraphs of G. Overall, the total time for constructing the tree T is in $O(pm) = \tilde{O}(n^2/D)$.

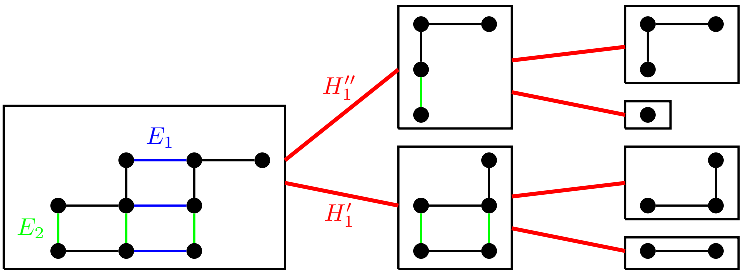


Figure 5 An example of tree T associated with a graph G for $D = 3$: here, $p = 2$.

Then, we compute the list of eccentricities for all the subgraphs labelling a node, by dynamic programming on T. In particular, doing so we compute the list of eccentricities for G because it is the label of the root. There are two cases:

- If H labels a leaf (base case) then, we claim that we have $\dim(H) \leq \lfloor \log D \rfloor + 1$. Indeed, by Lemma 4.18, every Θ-class of H is contained in a Θ-class of G. Since we removed all Θ-classes of G with at least D edges, the claim now follows from Lemma 4.20. In particular, we can compute the list of all eccentricities for H in $\tilde{O}(c^{\lfloor \log D \rfloor + 1}|V(H)|) = \tilde{O}(D^{\log c}|V(H)|)$ time. Recall that the leaves of T partition $V(G)$, and therefore, the total runtime for computing the list of eccentricities for the leaves is in $\tilde{O}(D^{\log c}n)$.
- From now on, let us assume H labels an internal node of T (inductive case). Let H'_i, H''_i be its children nodes, obtained from the removal of $E(H) \cap E_i$ for some $1 \leq i \leq p$. – For convenience, we will say later in the proof that H is an i-node. – Recall that $E(H) \cap E_i$ is a Θ-class of H. In particular, H'_i, H''_i are gated subgraphs. By Lemma 4.17, we can compute in $O(|V(H'_i)|)$ time the eccentricities in H of all vertices in H'_i if we are given as input: the list of eccentricities in H'_i, the list of eccentricities in H''_i, and for every $v \in V(H'_i)$ its gate $v^* \in \partial H''_i$ and the distance $d(v, v^*)$. The respective lists of eccentricities for H'_i and H''_i were pre-computed by dynamic programming on T. Furthermore, we can compute the gate v^* and $d(v, v^*)$ for every vertex $v \in V(H'_i)$, in total $\tilde{O}(|V(H)|)$ time, by using a modified BFS rooted at H''_i (we refer to [22, Lemma 17] for a detailed description of this standard procedure). Overall (by proceeding the same way for H''_i as for H'_i) we can compute the list of eccentricities for H in $\tilde{O}(|V(H)|)$ time. This is in total $\tilde{O}(n)$ time for the i-nodes (*i.e.*, because they were leaves of T at step i, and therefore, they are vertex-disjoint), and so, in total $\tilde{O}(pn) = \tilde{O}(n^2/D)$ time for all the internal nodes.

The total runtime for our algorithm is $\tilde{O}(n^2/D + D^{\log c}n)$, and optimized for $D = n^{\frac{1}{\log c + 1}}$. ◀

▶ **Theorem 4.22.** *There is an $\tilde{O}(n^{5/3})$-time algorithm for computing all eccentricities in median graphs.*

Proof. This result directly follows from Theorem 4.16 with Lemma 4.21 (for $c = 4$). ◀

As a natural extension of this work, the question of designing a linear-time or quasilinear-time algorithm to compute the diameter and all eccentricities of median graphs is now open. With the recursive splitting procedure of Lemma 4.21, unfortunately, the best execution time we could obtain is $\tilde{O}(n^{\frac{3}{2}})$. Reaching this bound could represent a first reasonable objective: it would "suffice" to propose a FPT combinatorial algorithm which computes all eccentricities in $\tilde{O}(2^d n)$ in order to obtain such time complexity.

Eventually, we note two lines of research on which this paper could have some influence: (i) the study of efficient algorithms for the computation of other metric parameters on median graphs (perhaps, the *betweenness centrality* [1]) and (ii) the design of subquadratic-time algorithms for the diameter and all eccentricities on larger families of graphs (*almost-median* or *semi-median* graphs [17, 30] for example).

References

1 A. Abboud, F. Grandoni, and V. V. Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. of SODA*, pages 1681–1697, 2015.

2 A. Abboud, V. V. Williams, and J. R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proc. of SODA*, pages 377–391, 2016.

3 S. P. Avann. Metric ternary distributive semi-lattices. *Proc. Amer. Math. Soc.*, 12:407–414, 1961.

4 H. Bandelt. Retracts of hypercubes. *Journal of Graph Theory*, 8(4):501–510, 1984.

5 H. Bandelt and V. Chepoi. Metric graph theory and geometry: a survey. *Contemp. Math.*, 453:49–86, 2008.

6 H. Bandelt, V. Chepoi, A. W. M. Dress, and J. H. Koolen. Combinatorics of lopsided sets. *Eur. J. Comb.*, 27(5):669–689, 2006.

7 H. Bandelt, V. Chepoi, and D. Eppstein. Combinatorics and geometry of finite and infinite squaregraphs. *SIAM J. Discret. Math.*, 24(4):1399–1440, 2010.

8 H. Bandelt, L. Quintana-Murci, A. Salas, and V. Macaulay. The fingerprint of phantom mutations in mitochondrial dna data. *Am. J. Hum. Genet.*, 71:1150–1160, 2002.

9 H. Bandelt and M. van de Vel. Embedding topological median algebras in products of dendrons. *Proc. London Math. Soc.*, 58:439–453, 1989.

10 H. J. Bandelt, P. Forster, B. C. Sykes, and M. B. Richards. Mitochondrial portraits of human populations using median networks. *Genetics*, 141(2):743–753, 1995.

11 J. Barthélemy and J. Constantin. Median graphs, parallelism and posets. *Discret. Math.*, 111(1-3):49–63, 1993.

12 J. Barthélemy, B. Leclerc, and B. Monjardet. On the use of ordered sets in problems of comparison and consensus of classifications. *Journal of Classification*, 3:187–224, 1986.

13 L. Bénéteau, J. Chalopin, V. Chepoi, and Y. Vaxès. Medians in median graphs and their cube complexes in linear time. In *Proc. of ICALP*, volume 168, pages 10:1–10:17, 2020.

14 P. Bergé, G. Ducoffe, and M. Habib. Subquadratic-time algorithm for the diameter and all eccentricities on median graphs. *CoRR*, abs/2110.02709, 2021.

15 P. Bergé and M. Habib. Diameter, radius and all eccentricities in linear time for constant-dimension median graphs. In *Proc. of LAGOS*, 2021.

16 G. Birkhoff and S. A. Kiss. A ternary operation in distributive lattices. *Bull. Amer. Math. Soc.*, 53:745–752, 1947.

17 B. Bresar. Characterizing almost-median graphs. *Eur. J. Comb.*, 28(3):916–920, 2007.

18 S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *Proc. of SODA*, pages 2143–2152, 2017.

19 S. Chechik, D. H. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. V. Williams. Better approximation algorithms for the graph diameter. In *Proc. of SODA*, pages 1041–1052, 2014.

20 V. Chepoi. Graphs of some CAT(0) complexes. *Adv. Appl. Math.*, 24(2):125–179, 2000.

21 V. Chepoi, F. F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *Proc. of SODA*, pages 346–355, 2002.

22 V. Chepoi, A. Labourel, and S. Ratel. Distance labeling schemes for cube-free median graphs. In *Proc. of MFCS*, volume 138, pages 15:1–15:14, 2019.

23 G. Ducoffe. Isometric embeddings in trees and their use in distance problems. In *Proc. of MFCS*, volume 202 of *LIPIcs*, pages 43:1–43:16, 2021.

24 G. Ducoffe, M. Habib, and L. Viennot. Diameter computation on H-minor free graphs and graphs of bounded (distance) VC-dimension. In *Proc. of SODA*, pages 1905–1922, 2020.

25 J. Hagauer, W. Imrich, and S. Klavzar. Recognizing median graphs in subquadratic time. *Theor. Comput. Sci.*, 215(1-2):123–136, 1999.

26 R. Hammack, W. Imrich, and S. Klavzar. *Handbook of Product Graphs, Second Edition*. CRC Press, Inc., 2011.

27 W. Imrich, S. Klavzar, and H. M. Mulder. Median graphs and triangle-free graphs. *SIAM J. Discret. Math.*, 12(1):111–118, 1999.

28 S. Klavzar and H. M. Mulder. Partial cubes and crossing graphs. *SIAM J. Discret. Math.*, 15(2):235–251, 2002.

29 S. Klavzar, H. M. Mulder, and R. Skrekovski. An Euler-type formula for median graphs. *Discret. Math.*, 187(1-3):255–258, 1998.

30 S. Klavzar and S. V. Shpectorov. Characterizing almost-median graphs II. *Discret. Math.*, 312(2):462–464, 2012.

31 M. Kovse. Complexity of phylogenetic networks: counting cubes in median graphs and related problems. *Analysis of complex networks: From Biology to Linguistics*, pages 323–350, 2009.

32 F. R. McMorris, H. M. Mulder, and F. S. Roberts. The median procedure on median graphs. *Discret. Appl. Math.*, 84(1-3):165–181, 1998.

33 H. M. Mulder and A. Schrijver. Median graphs and Helly hypergraphs. *Discret. Math.*, 25(1):41–50, 1979.

34 M. Mulder. The structure of median graphs. *Discret. Math.*, 24(2):197–204, 1978.

35 M. Mulder. The interval function of a graph. *Mathematical Centre Tracts, Mathematisch Centrum, Amsterdam*, 1980.

36 L. Roditty and V. V. Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. of STOC*, pages 515–524, 2013.

37 V. Sassone, M. Nielsen, and G. Winskel. A classification of models for concurrency. In *Proc. of CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 82–96, 1993.

38 Peter M Winkler. Isometric embedding in products of complete graphs. *Discrete Applied Mathematics*, 7(2):221–225, 1984.

A Orthogonal Θ-classes and hypercubes

We present now another important notion on median graphs: *orthogonality*. In [31], Kovse studied a relationship between *splits* which refer to the halfspaces of Θ-classes. It says that two splits $\{H_i', H_i''\}$ and $\{H_j', H_j''\}$ are *incompatible* if the four sets $H_i' \cap H_j'$, $H_i'' \cap H_j'$, $H_i' \cap H_j''$, and $H_i'' \cap H_j''$ are nonempty. Another definition was proven equivalent to this one.

▶ **Definition A.1** (Orthogonal Θ-classes). *We say that classes E_i and E_j are* orthogonal *($E_i \perp E_j$) if there is a square $uvyx$ in G, where $uv, xy \in E_i$ and $ux, vy \in E_j$.*

Indeed, classes E_i and E_j are orthogonal if and only if the splits produced by their halfspaces are incompatible.

▶ **Lemma A.2** (Orthogonal⇔Incompatible [15]). *Given two Θ-classes E_i and E_j of a median graph G, the following statements are equivalent:*

- *Classes E_i and E_j are orthogonal,*
- *Splits $\{H'_i, H''_i\}$ and $\{H'_j, H''_j\}$ are incompatible,*
- *The four sets $\partial H'_i \cap \partial H'_j$, $\partial H''_i \cap \partial H'_j$, $\partial H'_i \cap \partial H''_j$, and $\partial H''_i \cap \partial H''_j$ are nonempty.*

The concept of *orthogonality* is sometimes described with different words in the literature depending on the context: incompatible, *concurrent* or *crossing*. We say that E_i and E_j are *parallel* if they are not orthogonal, that is $H_i \subseteq H_j$ for some $H_i \in \{H'_i, H''_i\}$ and $H_j \in \{H'_j, H''_j\}$.

We pursue with a property on orthogonal Θ-classes: if two edges of two orthogonal classes E_i and E_j are incident, they belong to a common square.

▶ **Lemma A.3** (Squares [11, 15]). *Let $xu \in E_i$ and $uy \in E_j$. If E_i and E_j are orthogonal, then there is a vertex v such that $uyvx$ is a square.*

Pairwise orthogonal families. We focus on the set of Θ-classes which are pairwise orthogonal.

▶ **Definition A.4** (Pairwise Orthogonal Family). *We say that a set of classes $X \subseteq \mathcal{E}$ is a* Pairwise Orthogonal Family (POF for short) *if for any pair $E_j, E_h \in X$, we have $E_j \perp E_h$.*

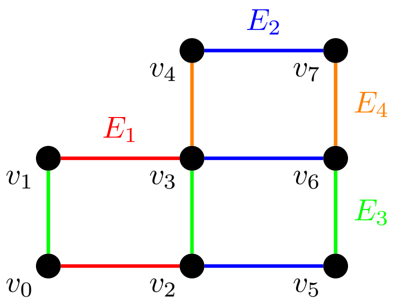
This notion is not completely new, since it implicitely appears in certain properties established on median graphs (for instance, the *downward cube* property in [13]). The empty set is considered as a POF. We denote by $\mathcal{L}$ the set of POFs of the median graph G. The notion of POF is strongly related to the induced hypercubes in median graphs. First, observe that all Θ-classes of a median graph form a POF if and only if the graph is a hypercube of dimension $\log n$ [31, 32]. Secondly, the next lemma precises the relationship between POFs and hypercubes.

▶ **Lemma A.5** (POFs adjacent to a vertex [15]). *Let X be a POF, $v \in V$, and assume that for each $E_i \in X$, there is an edge of E_i adjacent to v. There exists a hypercube Q containing vertex v and all edges of X adjacent to v. Moreover, the Θ-classes of the edges of Q are the classes of X.*

There is a natural bijection between the vertices of a median graph and the POFs. The next lemma exhibits this relationship.

▶ **Lemma A.6** (POFs and hypercubes [6, 8, 31]). *Consider an arbitrary canonical basepoint $v_0 \in V$ and the v_0-orientation for the median graph G. Given a vertex $v \in V$, let $N^-(v)$ be the set of edges going into v according to the v_0-orientation. Let $\mathcal{E}^-(v)$ be the classes of the edges in $N^-(v)$. The following propositions are true:*

- *For any vertex $v \in V$, $\mathcal{E}^-(v)$ is a POF. Moreover, vertex v and the edges of $N^-(v)$ belong to an induced hypercube formed by the classes $\mathcal{E}^-(v)$. Hence, $|\mathcal{E}^-(v)| = |N^-(v)| \leq d$.*
- *For any POF X, there is a unique vertex v_X such that $\mathcal{E}^-(v_X) = X$. Vertex v_X is the closest-to-v_0 vertex v such that $X \subseteq \mathcal{E}^-(v)$.*
- *The number of POFs in G is equal to the number n of vertices: $n = |\mathcal{L}|$.*

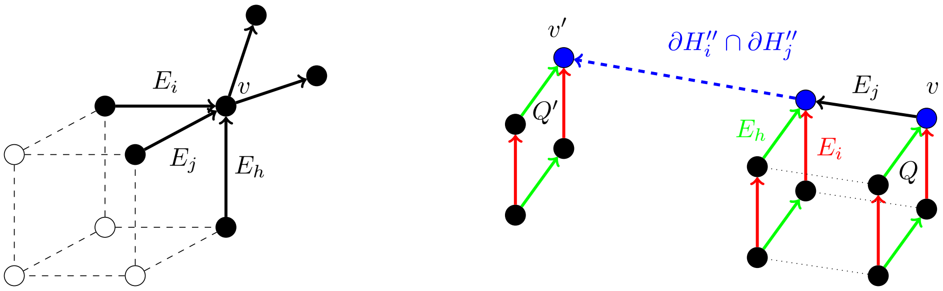


Vertex	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
POF	$\emptyset$	$\{E_3\}$	$\{E_1\}$	$\{E_1, E_3\}$	$\{E_4\}$	$\{E_2\}$	$\{E_2, E_3\}$	$\{E_2, E_4\}$

Figure 6 Illustration of the bijection between V and the set of POFs.

An example is given in Figure 6 with a small median graph of dimension $d = 2$. v_0 is the canonical basepoint and edges are colored according to their Θ-class. For example, $v_1v_3 \in E_1$. We associate with any POF X of G the vertex v_X satisfying $\mathcal{E}^-(v_X) = X$ with the v_0-orientation. Obviously, the empty POF is associated with v_0 which has no incoming edges.

A straightforward consequence of this bijection is that parameter q, the number of Θ-classes, is less than the number of vertices n. But it can be used less trivially to enumerate the POFs of a median graph in linear time [8, 31]. Given a basepoint v_0, we say that the *basis* (resp. *anti-basis*) of an induced hypercube Q is the single vertex v such that all edges of the hypercube adjacent to v are outgoing from (resp. incoming into) v. Said differently, the basis of Q is its closest-to-v_0 vertex and its anti-basis is its farthest-to-v_0 vertex. What Lemma A.6 states is also that we can associate with any POF X a hypercube Q_X which contains exactly the classes X and admits v_X as its anti-basis. This observation implies that the number of POFs is less than the number of hypercubes in G. Moreover, the hypercube Q_X is the closest-to-v_0 hypercube formed with the classes in X. Figure 7a shows a vertex v with its incoming and outgoing edges with the v_0-orientation. The dashed edges represent the hypercube with anti-basis v and POF $\mathcal{E}^-(v)$.



(a) The hypercube “induced” by the edges incoming into a vertex (its antibasis).

(b) A POF signing at least two hypercubes Q and Q' is not maximal.

Figure 7 Properties of POFs.

Number of hypercubes. We remind a formula establishing a relationship between the number of POFs and the number of hypercubes in the literature. Let $\alpha(G)$ (resp. $\beta(G)$) be the number of hypercubes (resp. POFs) in G. Let $\beta_i(G)$ be the number of POFs of cardinality $i \leq d$ in G. According to [8, 31], we have:

$$\alpha(G) = \sum_{i=0}^{d} 2^i \beta_i(G). \tag{4}$$

Equation (4) produces a natural upper bound for the number of hypercubes.

▶ **Lemma A.7** (Number of hypercubes). *$\alpha(G) \leq 2^d n$.*

Value $\alpha(G)$ consider all hypercubes, in particular those of dimension 0, *i.e.* vertices. From now on, the word "hypercube" refers to the hypercubes of dimension at least one.

Each hypercube in the median graph G can be defined with only its anti-basis v and the edges $\widehat{N}$ of the hypercube that are adjacent and going into v according to the v_0-orientation. These edges are a subset of $N^-(v)$: $\widehat{N} \subseteq N^-(v)$. Conversely, given a vertex v, each subset of $N^-(v)$ produces a hypercube which admits v as an anti-basis (this hypercube is a sub-hypercube of the one obtained with v and $N^-(v)$, Lemma A.6). Another possible bijection is to consider a hypercube as a pair composed of its anti-basis v and the Θ-classes $\widehat{\mathcal{E}}$ of the edges in $\widehat{N}$ (its signature).

As a consequence, a simple graph search as BFS enables us to enumerate the hypercubes in G in time $O(d2^d n)$.

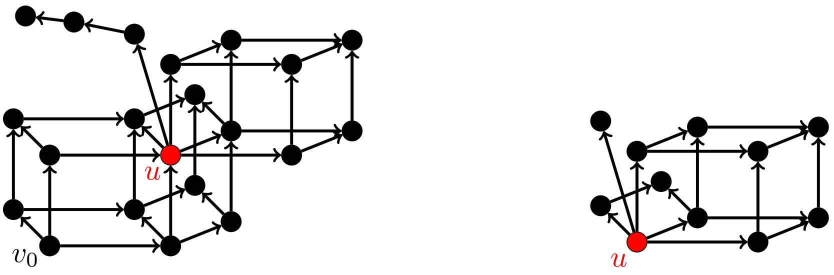
▶ **Lemma A.8** (Enumeration of hypercubes [15]). *We can enumerate all triplets $(v, u, \widehat{\mathcal{E}})$, where v is the anti-basis of a hypercube Q, u its basis, and $\widehat{\mathcal{E}}$ the signature of Q in time $O(d2^d n)$. Moreover, the list obtained fulfils the following partial order: if $d(v_0, v) < d(v_0, v')$, then any triplet $(v, u, \widehat{\mathcal{E}})$ containing v appears before any triplet $(v', u', \widehat{\mathcal{E}}')$ containing v'.*

The enumeration of hypercubes is thus executed in linear time for median graphs with constant dimension. In summary, given any median graph, one can compute the set of Θ-classes and their orthogonality relationship (for each E_i, the set of Θ-classes orthogonal to E_i) in linear time, and the set of hypercubes with its basis, anti-basis and signature in $\tilde{O}(2^d n)$.

B Proofs of Section 4

▶ **Theorem B.1** (Computation of labels op). *There is a combinatorial algorithm which determines all labels $\mathsf{op}_u(L)$ in $\tilde{O}(2^d n)$.*

Proof. Let $u \in V$: we denote by N_u the number of hypercubes of G with basis u. Convex subgraphs of median graphs are also median by considering the original definition of median graphs (Definition 2.1). Consequently, star graph G_u is median and all its maximal hypercubes contain a common vertex u. From Theorem 3.2, G_u is a simplex graph.



(a) A v_0-oriented median graph G and a vertex $u \in V$. **(b)** Star graph G_u.

Figure 8 Example of star graph G_u.

Any pair (u, L) of G, where L is a POF outgoing from u in G, can be associated to a unique hypercube with signature L and basis u. Thus, there is a natural bijection between (i) the POFs of G_u (ii) the vertices of G_u and (iii) the POFs L of G outgoing from u. Hence, $|V_u| = N_u$.

We associate with any POF L of G_u the weight $\omega_u(L) = \varphi(u, L)$. We apply the algorithm evoked in Theorem 3.8. The opposite computed with that configuration correspond exactly to the labels $\mathsf{op}_u(L)$: a POF L' disjoint from L and maximizing $\varphi(u, L')$ among all POFs outgoing from u. The running time of the algorithm is $O((d^3 + \log |V_u|)\,|V_u|) = O((d^3 + \log n)N_u)$. Doing it for every vertex u of G, we obtain all opposite labels of G in $O(d^3 + \log n)2^d n)$ as $\sum_{u \in V} N_u = 2^d n$ (Lemma A.7). ◀

▶ **Lemma B.2.** *Let G be a median graph. For every $1 \leq i \leq q$, let $v \in V(H_i')$ be arbitrary, and let v^* be its gate in $\partial H_i''$. Then, $\mathsf{ecc}(v) = \max\{\mathsf{ecc}_{H_i'}(v), d(v, v^*) + \mathsf{ecc}_{H_i''}(v^*)\}$.*

Proof. We have $\mathsf{ecc}(v) = \mathsf{ecc}_G(v) = \max\{d(u, v) \mid u \in V(H_i')\} \cup \{d(w, v) \mid w \in V(H_i'')\}$. Since H_i' is convex, we have $\max\{d(u, v) \mid u \in V(H_i')\} = \mathsf{ecc}_{H_i'}(v)$. In the same way, since H_i'' is gated (and so, convex), we have $\max\{d(w, v) \mid w \in V(H_i'')\} = d(v, v^*) + \max\{d(v^*, w) \mid w \in V(H_i'')\} = d(v, v^*) + \mathsf{ecc}_{H_i''}(v^*)$. ◀

▶ **Lemma B.3.** *Let H and G be median graphs. If H is an induced subgraph of G then, every Θ-class of H is contained in a Θ-class of G.*

Proof. Every square of H is also a square of G. In particular, two edges of H are in relation Θ_0 if and only if, as edges of G, they are also in relation Θ_0. Since the Θ-classes of H (resp., of G) are the transitive closure of its relation Θ_0, it follows that every Θ-class of H is contained in a Θ-class of G. ◀

▶ **Lemma B.4.** *Let H and G be median graphs, and let $E_1, E_2, \ldots, E_q$ denote the Θ-classes of G. If H is an isometric subgraph of G then, the Θ-classes of H are exactly the nonempty subsets among $E_i \cap E(H)$, for $1 \leq i \leq q$.*

Proof. It is known [38] that two edges uv, xy of G are in the same Θ-class if and only if $d_G(u, x) + d_G(v, y) \neq d_G(u, y) + d_G(v, x)$. In particular, since H is isometric in G, two edges of H are in the same Θ-class of H if and only if they are in the same Θ-class of G. ◀

Faster Counting and Sampling Algorithms Using Colorful Decision Oracle

Anup Bhattacharya ✉ 🏠
National Institute of Science Education and Research, Bhubaneswar, India

Arijit Bishnu ✉ 🏠
Indian Statistical Institute, Kolkata, India

Arijit Ghosh ✉ 🏠
Indian Statistical Institute, Kolkata, India

Gopinath Mishra ✉ 🏠
University of Warwick, Coventry, UK

Abstract

In this work, we consider d-Hyperedge Estimation and d-Hyperedge Sample problem in a hypergraph $\mathcal{H}(U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$ in the query complexity framework, where $U(\mathcal{H})$ denotes the set of vertices and $\mathcal{F}(\mathcal{H})$ denotes the set of hyperedges. The oracle access to the hypergraph is called Colorful Independence Oracle (CID), which takes d (non-empty) pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ as input, and answers whether there exists a hyperedge in $\mathcal{H}$ having (exactly) one vertex in each $A_i, i \in \{1, 2, \ldots, d\}$. The problem of d-Hyperedge Estimation and d-Hyperedge Sample with CID oracle access is important in its own right as a combinatorial problem. Also, Dell et al. [SODA '20] established that *decision* vs *counting* complexities of a number of combinatorial optimization problems can be abstracted out as d-Hyperedge Estimation problems with a CID oracle access.

The main technical contribution of the paper is an algorithm that estimates $m = |\mathcal{F}(\mathcal{H})|$ with $\widehat{m}$ such that

$$\frac{1}{C_d \log^{d-1} n} \le \frac{\widehat{m}}{m} \le C_d \log^{d-1} n.$$

by using at most $C_d \log^{d+2} n$ many CID queries, where n denotes the number of vertices in the hypergraph $\mathcal{H}$ and C_d is a constant that depends only on d. Our result coupled with the framework of Dell et al. [SODA '21] implies improved bounds for the following fundamental problems:

Edge Estimation using the Bipartite Independent Set (BIS). We improve the bound obtained by Beame et al. [ITCS '18, TALG '20].

Triangle Estimation using the Tripartite Independent Set (TIS). The previous best bound for the case of graphs with low *co-degree* (Co-degree for an edge in the graph is the number of triangles incident to that edge in the graph) was due to Bhattacharya et al. [ISAAC '19, TOCS '21], and Dell et al.'s result gives the best bound for the case of general graphs [SODA '21]. We improve both of these bounds.

Hyperedge Estimation & Sampling using Colorful Independence Oracle (CID). We give an improvement over the bounds obtained by Dell et al. [SODA '21].

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Query Complexity, Subset Query, Hyperedge Estimation, and Colorful Independent Set oracle

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.10

Related Version *Full Version*: `https://arxiv.org/abs/2201.04975`

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 10; pp. 10:1–10:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Estimating different combinatorial structures like edges, triangles and cliques in an unknown graph that can be accessed only through *query oracles* is a fundamental area of research in *sublinear algorithms* [13, 14, 11, 12]. Different query oracles provide unique ways of looking at the same graph. Beame et al. [1] introduced an independent set based subset query oracle, named Bipartite Independent Set (BIS) query, to estimate the number of edges in a graph using polylogarithmic queries. The BIS query answers a YES/NO question on the existence of an edge between two disjoint subsets of vertices of a graph G. The next natural questions in this line of research were problems of estimation and uniform sampling of hyperedges in hypergraphs [9, 3, 4]. In this paper, we will be focusing on these two fundamental questions, and in doing so, we will improve all the previous results [2, 9, 3, 4].

1.1 Our query oracle, results and the context

A hypergraph $\mathcal{H}$ is a *set system* $(U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$, where $U(\mathcal{H})$ denotes a set of n vertices and $\mathcal{F}(\mathcal{H})$, a set of subsets of $U(\mathcal{H})$, denotes the set of hyperedges. A hypergraph $\mathcal{H}$ is said to be d-*uniform* if every hyperedge in $\mathcal{H}$ consists of exactly d vertices. The cardinality of the hyperedge set is denoted as $m(\mathcal{H}) = |\mathcal{F}(\mathcal{H})|$. We will access the hypergraph using the following oracle[1] [6].

▶ **Definition 1.1** (Colorful Independent Set (CID)). *Given d pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ as input,* CID *query answers* Yes *if and only if $m(A_1, \ldots, A_d) \neq 0$, where $m(A_1, \ldots, A_d)$ denotes the number of hyperedges in $\mathcal{H}$ having exactly one vertex in each A_i, where $i \in \{1, 2, \ldots, d\}$.*

Note that the earlier mentioned BIS is a special case of CID when $d = 2$. With this query oracle access, we solve the following two problems.

d-Hyperedge-Estimation
Input: Vertex set $U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ with n vertices, a CID oracle access to $\mathcal{H}$, and $\varepsilon \in (0, 1)$.
Output: A $(1 \pm \varepsilon)$-approximation $\widehat{m}$ to $m(\mathcal{H})$ with probability $1 - 1/n^{\Omega(d)}$.

Note that Edge Estimation problem is a special case of d-Hyperedge-Estimation when $d = 2$.

d-Hyperedge-Sample
Input: Vertex set $U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ with n vertices, a CID oracle access to $\mathcal{H}$, and $\varepsilon \in (0, 1)$.
Output: With probability $1 - 1/n^{\Omega(d)}$, report a sample from a distribution of hyperedges in $\mathcal{H}$ such that the probability that any particular hyperedge is sampled lies in the interval $\left[(1-\varepsilon)\frac{1}{m}, (1+\varepsilon)\frac{1}{m}\right]$.

This area started with the investigation of Edge Estimation problem by Dell and Lapinskas [7, 8] and Beame et al. [1], then Bhattacharya et al. [3, 4] studied d-Hyperedge-Estimation for $d = 3$, and more recently Dell et al. [9] gave algorithms for d-Hyperedge-Estimation and d-Hyperedge-Sample for general d. Beame et al. [1] showed that Edge

[1] In [6], the oracle is named as Generalized Partite Independent Set oracle. Here, we follow the same suit as Dell et al. [9] with respect to the name of the oracle.

Estimation problem can be solved using $\mathcal{O}\left(\frac{\log^{14} n}{\varepsilon^4}\right)$ BIS queries. Having estimated the number of edges in a graph using BIS queries, a very natural question was to estimate the number of hyperedges in a hypergraph using an appropriate query oracle. This extension is nontrivial as two edges in a graph can intersect in at most one vertex but the intersection pattern between two hyperedges in a hypergraph is more complicated. As a first step towards resolving this question, Bhattacharya et al. [3, 4] considered d-Hyperedge-Estimation in 3-uniform hypergraphs using CID queries. They showed that when *co-degree* of any pair of vertices in a 3-uniform hypergraph is bounded above by Δ, then one can solve d-Hyperedge-Estimation using $\mathcal{O}\left(\frac{\Delta^2 \log^{18} n}{\varepsilon^4}\right)$ CID queries. Recall that co-degree of two vertices in a hypergraph is the number of hyperedges that contain both vertices. Dell et al. [9] generalized the results of Beame et al. [1] and Bhattacharya et al. [3, 4], and obtained a similar (with an improved dependency in terms of ε) result for the d-Hyperedge-Estimation problem for general d. Apart from d-Hyperedge-Estimation problem, they also considered the problem of d-Hyperedge-Sample. The results of Dell et al. [9] are formally stated in the following proposition:

▶ **Proposition 1.2** (Dell et al. [9]). *d-Hyperedge-Estimation and d-Hyperedge-Sample can be solved by using $\mathcal{O}_d\left(\frac{\log^{4d+8} n}{\varepsilon^2}\right)$ and $\mathcal{O}_d\left(\frac{\log^{4d+12} n}{\varepsilon^2}\right)$ CID queries, respectively.* [2]

Currently, the best known bound (prior to this work) for solving d-Hyperedge-Estimation problem, for general d, is due to Dell et al. [9], but note that for constant $\varepsilon \in (0, 1)$, Beame et al. [1, 2] still have the best bound for the Edge Estimation problem.

Our main result is an improved *coarse estimation* technique, named Rough Estimation, and is stated in the following theorem. The significance of the coarse estimation technique will be discussed in Section 1.2.

▶ **Theorem 1.3** (Main result). *There exists an algorithm* Rough Estimation *that has* CID *query access to a d-uniform hypergraph $\mathcal{H}(U, \mathcal{F})$ and returns $\widehat{m}$ as an estimate for $m = |\mathcal{F}(\mathcal{H})|$ such that*

$$\frac{1}{C_d \log^{d-1} n} \leq \frac{\widehat{m}}{m} \leq C_d \log^{d-1} n$$

with probability at least $1 - 1/n^{\Omega(d)}$ using at most $C_d \log^{d+2} n$ CID *queries, where C_d is a constant that depends only on d and n denotes the number of vertices in $\mathcal{H}$.*

Coarse estimation gives a crude polylogarithmic approximation for m, the number of hyperedges in $\mathcal{H}$. This improvement in the coarse estimation algorithm coupled with *importance sampling* and the algorithmic framework of Dell et al. [9] gives an improved algorithm for both d-Hyperedge-Estimation and d-Hyperedge-Sample problems.

▶ **Theorem 1.4** (Improved bounds for estimating and sampling). *d-Hyperedge-Estimation and d-Hyperedge-Sample problems can be solved by using $\mathcal{O}_d\left(\frac{\log^{3d+5} n}{\varepsilon^2}\right)$ and $\mathcal{O}_d\left(\frac{\log^{3d+9} n}{\varepsilon^2}\right)$* CID *queries, respectively.*

[2] Dell et al. [9] studied d-Hyperedge-Estimation and d-Hyperedge-Sample where the probability of success is $1 - \delta$ for some given $\delta \in (0, 1)$, and have showed that d-Hyperedge-Estimation and d-Hyperedge-Sample can be solved by using $\mathcal{O}_d\left(\frac{\log^{4d+7} n}{\varepsilon^2} \log \frac{1}{\delta}\right)$ and $\mathcal{O}_d\left(\frac{\log^{4d+11} n}{\varepsilon^2} \log \frac{1}{\delta}\right)$ CID queries, respectively. In Proposition 1.2, we have taken $\delta = n^{\mathcal{O}(d)}$. But both the results of Beame et al. [1, 2] and Bhattacharya et al. [3, 4] are in the high probability regime.
In this paper, we work with success probability to be $1 - 1/n^{\Omega(d)}$ for simplicity of presentation and compare our results with all previous results in a high probability regime.

The details regarding how Theorem 1.3 can be used together with the framework of Dell et al. [9] to prove Theorem 1.4 will be discussed in Section 5.

Using Theorem 1.4, we directly get the following improved bounds for Edge Estimation and d-Hyperedge-Estimationin 3-uniform hypergraph by substituting $d = 2$ and $d = 3$, respectively.

▶ **Corollary 1.5.**

(a) Edge Estimation *can be solved using* $\mathcal{O}\left(\frac{\log^{11} n}{\varepsilon^2}\right)$ *queries to* Bipartite Independent Set *(*BIS*) oracle.*

(b) d-Hyperedge-Estimation *in a* 3*-uniform hypergraph can be solved using* $\mathcal{O}\left(\frac{\log^{14} n}{\varepsilon^2}\right)$ CID *queries.*

The above corollary gives the best bound (till now) for the Edge Estimation. Recall that Bhattacharya et al. [3, 4] proved that when the co-degree of a 3-uniform graph is bounded by Δ then d-Hyperedge-Estimation in that hypergraph can be solved using $\mathcal{O}\left(\frac{\Delta^2 \log^{18} n}{\varepsilon^4}\right)$ CID queries. For fixed $\varepsilon \in (0, 1)$ and $\Delta = o(\log n)$ the bound obtained by Bhattacharya et al. [3, 4] is asymptotically better than the bound we get from Dell et al. [9], see Proposition 1.2. Note that Corollary 1.5 (b) improves the bounds obtained by Bhattacharya et al. [3, 4] and Dell et al. [9] for all values of Δ and $\varepsilon \in (0, 1)$.

1.2 Fundamental role of coarse estimation

The framework of Dell et al. [9] is inspired by the following observation. Let us consider $t = \mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ independent subhypergraphs each induced by $n/2$ uniform random vertices. The probability, that a particular hyperedge is present in a subhypergraph induced by $n/2$ many uniform random vertices, is $\frac{1}{2^d}$. Denoting X as the sum of the numbers of the hyperedges present in the t subhypergraphs, observe that $\frac{2^d}{t}X$ is a $(1 \pm \varepsilon)$-approximation of m. If we repeat the procedure recursively $\mathcal{O}(\log n)$ times, then all the subhypergraphs will have a bounded number of vertices in terms of d, at which point the number of hyperedges can be determined exactly by using $\mathcal{O}_d(1)$ CID queries. However, the number of induced subhypergraphs in the worst case can become as *large* as $\Omega\left((\log n)^{\log n}\right)$.

To have the number of subhypergraphs bounded at all point of time, they use *importance sampling*. It is about maintaining the weighted sum of some variables whose approximate value is known to us. The output will be a bounded number of variables and some weight parameters such that the weighted sum of the variables estimates the required sum. The objective of the importance sampling procedure in Beame et al. [1, 2] and Bhattacharya et al. [3, 4], are also the same [3]. However, Dell et al. improved the importance sampling result by the use of a particular form of Bernstein inequality and by a very careful analysis.

To apply importance sampling, it is required to have a rough estimate (possibly with a polylogarithmic approximation factor) of the number of hyperedges in each subhypergraph that are currently present for processing – this is what exactly coarse estimation does. The objective of coarse estimation in Beame et al. [1, 2] and Bhattacharya et al. [3, 4] are also the same [4]. But all these frameworks have a commonality. The approximation guarantee and the query complexity of the coarse estimation has a direct bearing on the query complexity of the final algorithm.

[3] In fact, Bhattacharya et al. [3, 4] directly use the importance sampling developed by Beame et al. [1, 2]

[4] Note that the main merit of the framework of Dell et al. [9] over Beame et al. [1, 2] and Bhattacharya et al. [3, 4] is not only that it generalized to hypergraph, but also the dependence on ε is $1/\varepsilon^2$ in Dell et al. [9]'s work as opposed to $\frac{1}{\varepsilon^4}$ in Beame et al. [1, 2] and Bhattacharya et al. [3, 4].

Therefore, any improvement in the coarse estimation algorithm will directly improve the query complexities of d-Hyperedge-Estimation and d-Hyperedge-Sample. In this paper, we focus on improving the coarse estimation algorithm.

1.3 Setup and notations

We denote the sets $\{1,\ldots,n\}$ and $\{0,\ldots,n\}$ by $[n]$ and $[n^*]$, respectively. A hypergraph $\mathcal{H}$ is a *set system* $(U(\mathcal{H}),\mathcal{F}(\mathcal{H}))$, where $U(\mathcal{H})$ denotes the set of vertices and $\mathcal{F}(\mathcal{H})$ denotes the set of hyperedges. The set of vertices present in a hyperedge $F \in \mathcal{F}(\mathcal{H})$ is denoted by $U(F)$ or simply F. A hypergraph $\mathcal{H}$ is said to be d-uniform if all the hyperedges in $\mathcal{H}$ consist of exactly d vertices. The cardinality of the hyperedge set is $m(\mathcal{H}) = |\mathcal{F}(\mathcal{H})|$. For $A_1,\ldots,A_d \subseteq U(\mathcal{H})$ (not necessarily pairwise disjoint), $\mathcal{F}(A_1,\ldots,A_d) \subseteq \mathcal{F}(\mathcal{H})$ denotes the set of hyperedges having a vertex in each A_i, and $m(A_1,\ldots,A_d)$ is the number of hyperedges in $|\mathcal{F}(A_1,\ldots,A_d)|$.

Let $\mathbb{E}[X]$ and $\mathbb{V}[X]$ denote the expectation and variance of the random variable X. For an event $\mathcal{E}$, the complement of $\mathcal{E}$ is denoted by $\overline{\mathcal{E}}$. The statement "a is a $(1\pm\varepsilon)$-approximation of b" means $|b-a| \leq \varepsilon \cdot b$. For $x \in \mathbb{R}$, $\exp(x)$ denotes the standard exponential function e^x. In this paper, d is a constant, and $\mathcal{O}_d(\cdot)$ and $\Omega_d(\cdot)$ denote the standard $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$, where the constant depends only on d. We use $\log^k n$ to denote $(\log n)^k$. By polylogarithmic, we mean $\mathcal{O}_d\left(\frac{\log^{\mathcal{O}(d)} n}{\varepsilon^{\Omega(1)}}\right)$ in this paper.

1.4 Paper organization

In Section 2, we describe the notion of an *ordered hyperedge*, and define three other query oracles that can be simulated by using $\mathcal{O}_d(\log n)$ CID queries. The role of ordered hyperedges and these oracles are mostly expository purposes, i.e., they help us to describe our algorithms and the calculations more neatly. Section 3 gives a brief overview of the proof of our main technical result. In Section 4 we give the proof of our main result (Theorem 1.3). We describe in Section 5 implications of our main result and how Theorem 1.3 can be used to prove Theorem 1.4. The equivalence proofs of the CID oracle and its variants are discussed in Section 2. Some useful probability results are given in Appendix A. Since we use different types of oracles in the calculations, we have recalled all their definitions in Appendix B for the ease of reference. Proofs omitted are marked with $\star$, and can be found in the full version [5] of this paper.

2 Preliminaries: Ordered hyperedges, CID oracle, and its variants

Ordered hyperedges

We will use the subscript "o" to denote the set of ordered hyperedges. For example, $\mathcal{H}_o(U,\mathcal{F}_o)$ denotes the ordered hypergraph corresponding to $\mathcal{H}(U,\mathcal{F})$. Here $\mathcal{F}_o(\mathcal{H})$ denotes the set of ordered hyperedges that contains $d!$ ordered d-tuples for each hyperedge in $\mathcal{H}(U,\mathcal{F})$. Let $m_o(\mathcal{H}_o)$ denotes $|\mathcal{F}_o(\mathcal{H}_o)|$. Note that $m_o(\mathcal{H}_o) = d!m(\mathcal{H})$. Also, let $\mathcal{F}_o(A_1,\ldots,A_d)$ denotes the set $\{F_o \in \mathcal{F}_o(\mathcal{H}) : \text{the } i\text{-th vertex of } F_o \text{ is in } A_i, \forall i \in [d]\}$. The corresponding number for ordered hyperedges is $m_o(A_1,\ldots,A_d)$. Note that $\mathcal{F}_o(U(\mathcal{H}),\ldots,U(\mathcal{H})) = \mathcal{F}_o(\mathcal{H})$.

CID oracle and its variants

Note that the CID query takes as input d pairwise disjoint subsets of vertices. We now define two related query oracles CID$_1$ and CID$_2$ that remove the disjointness requirements for the input. Then we extent CID$_2$ to the ordered setting. We show that both query oracles can be simulated, with high probability, by making $\mathcal{O}_d(\log n)$ queries to the CID oracle. The oracles CID_1 and CID_2 will be used in the description of the algorithm for ease of exposition.

CID_1: Given s pairwise disjoint subsets of vertices $A_1, \ldots, A_s \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ and $a_1, \ldots, a_s \in [d]$ such that $\sum_{i=1}^{s} a_i = d$, CID_1 query on input $A_1^{[a_1]}, A_2^{[a_2]}, \cdots, A_s^{[a_s]}$ answers YES if and only if $m(A_1^{[a_1]}, \ldots, A_s^{[a_s]}) \neq 0$. Here $A^{[a]}$ denotes the set A repeated a times.

CID_2: Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$, CID_2 query on input $A_1, \ldots, A_d$ answers YES if and only if $m(A_1, \ldots, A_d) \neq 0$.

CID_2^o: Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H}_o)$ of an ordered hypergraph $\mathcal{H}_o$, CID_2^o query on input $A_1, \ldots, A_d$ answers YES if and only if $m_o(A_1, \ldots, A_d) \neq 0$.

Observe that the CID_2 query is the same as the CID query without the requirement that the input sets are disjoint. For the CID_1 query, multiple repetitions of the same set is allowed in the input. It is obvious that a CID query can be simulated by a CID_1 or CID_2 query. Also, CID_2^o is the ordered analogue of CID_2 . Using the following observation, we show how a CID_2^o, CID_1 , or a CID_2 query can be simulated by a polylogarithmic number of CID queries.

▶ **Observation 2.1** ($\star$, Connection between query oracles). *Let $\mathcal{H}(U, \mathcal{F})$ denote a hypergraph and $\mathcal{H}_o(U, \mathcal{F}_o)$ denote the corresponding ordered hypergraph.*

(i) *A CID_1 query to $\mathcal{H}(U, \mathcal{F})$ can be simulated using $\mathcal{O}_d(\log n)$ CID queries with probability $1 - 1/n^{\Omega(d)}$.*

(ii) *A CID_2 query $\mathcal{H}(U, \mathcal{F})$ can be simulated using $\mathcal{O}_d(1)$ CID_1 queries.*

(iii) *A CID_2 query $\mathcal{H}(U, \mathcal{F})$ can be simulated using $\mathcal{O}_d(\log n)$ CID queries with probability $1 - 1/n^{\Omega(d)}$.*

(iv) *A CID_2^o query to $\mathcal{H}_o(U, \mathcal{F}_o)$ can be simulated using a CID_2 query to $\mathcal{H}(U, \mathcal{F})$.*

3 Overview of the main structural result

To prove Theorem 1.3, we first consider Lemma 3.1, which is the central result of the paper and is the ordered hypergraph analogue of Theorem 1.3. The main theorem (Theorem 1.3) follows from Lemma 3.1 along with Observation 2.1.

▶ **Lemma 3.1** (Main Lemma). *There exists an algorithm* Rough Estimation *that has CID_2^o query access to a d-uniform ordered hypergraph $\mathcal{H}_o(U, \mathcal{F}_o)$ corresponding to hypergraph $\mathcal{H}(U, \mathcal{F})$ and returns $\widehat{m}_o$ as an estimate for $m_o = |\mathcal{F}_o(\mathcal{H}_o)|$ such that*

$$\frac{1}{C_d \log^{d-1} n} \leq \frac{\widehat{m}}{m} \leq C_d \log^{d-1} n$$

with probability at least $1 - 1/n^{\Omega(d)}$ using at most $C_d \log^{d+1} n$ CID_2^o queries, where C_d is a constant that depends only on d.

At a high level, the idea for an improved coarse estimation involves a recursive bucketing technique and careful analysis of the intersection pattern of hypergraphs.

To build up towards the final proof, we need to prove Lemma 3.1. Towards this end, we first define some quantities and prove Claim 3.2. For that, let us think of partitioning the vertex set in $U_1 = U(\mathcal{H})$ into buckets such that the vertices in each bucket appear as the first vertex in approximately the same number of hyperedges. So, there will be at most $d \log n + 1$ buckets. It can be shown that that there is a bucket $Z_1 \subseteq U_1$ such that the number of hyperedges, having the vertices in the bucket as the first vertex, is at least $\frac{m_o}{d \log n + 1}$. For each vertex $z_1 \in Z_1$, let the number of hyperedges in $\mathcal{H}_o$, having z_1 as the first vertex, lie between 2^{q_1} and $2^{q_1+1} - 1$ for some suitable q_1. Then we can argue that

$$|Z_1| \geq \frac{m_o}{2^{q_1+1}(d\log n + 1)}.$$

Similarly, we extend the bucketing idea to tuples as follows. Consider a vertex a_1 in a particular bucket of U_1 and consider all the ordered hyperedges in $\mathcal{F}_o(a_1)$ containing a_1 as the first vertex. We can bucket the vertices in $U_2 = U(\mathcal{H})$ such that the vertices in each bucket of U_2 are present in approximately the same number of hyperedges in $\mathcal{F}_o(a_1)$ as the second vertex. We generalize the above bucketing strategy with the vertices in U_i's, which is formally described below. Notice that this way of bucketing will allow us to use conditionals on sampling vertices from the desired buckets of U_i's.

For $q_1 \in [(d\log n)^*]$, let $U_1(q_1) \subseteq U_1$ be the set of vertices in $a_1 \in U_1$ such that for each $a_1 \in U_1(q_1)$, the number of hyperedges in $\mathcal{F}_o(\mathcal{H}_o)$, containing a_1 as the first vertex, lies between 2^{q_1} and $2^{q_1+1}-1$. For $2 \leq i \leq d-1$, and $q_j \in [(d\log n)^*]$ for each $j \in [i-1]$, consider $a_1 \in U_1(q_1), a_2 \in U_2((q_1,a_1),q_2),\ldots,\ a_{i-1} \in U_{i-1}((q_1,a_1),\ldots,(q_{i-2},a_{i-2}),q_{i-1})$. Let $U_i((q_1,a_1),\ldots,(q_{i-1},a_{i-1}),q_i)$ be the set of vertices in U_i such that for each $u_i \in U_i((q_1,u_1),\ldots,(q_{i-1},a_{i-1}),q_i)$, the number of ordered hyperedges in $\mathcal{F}_o(\mathcal{H}_o)$, containing u_j as the j-th vertex for all $j \in [i]$, lies between 2^{q_i} and $2^{q_i+1}-1$. We need the following result to proceed further. For ease of presentation, we use (Q_i, A_i) to denote $(q_1,a_1),\ldots,(q_{i-1},a_{i-1})$ for $2 \leq i \leq d-1$. Informally, Claim 3.2 says that for each $i \in [d-1]$, there exists a bucket in U_i having a *large* number of vertices contributing approximately the same number of hyperedges..

▷ Claim 3.2 (⋆).

(i) There exists $q_1 \in [(d\log n)^*]$ such that

$$|U_1(q_1)| > \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d\log n + 1)}.$$

(ii) Let $2 \leq i \leq d-1$ and $q_j \in [(d\log n)^*]$ $\forall j \in [i-1]$. Let $a_1 \in U_1(q_1)$, $a_j \in U_j((Q_{j-1},A_{j-1}),q_j)$ $\forall j \neq 1$ and $j < i$. There exists $q_i \in [(d\log n)^*]$ such that

$$|U_i((Q_i,A_i),q_i)| > \frac{2^{q_{i-1}}}{2^{q_i+1}(d\log n + 1)}.$$

4 Proof of Lemma 3.1

We now prove Lemma 3.1 formally. The algorithm corresponding to Lemma 3.1 is Algorithm 2 (named Rough Estimation). Algorithm 1 (named Verify-Estimate) is a subroutine of Algorithm 2. Algorithm 1 determines whether a given estimate $\widehat{R}$ of the number of ordered hyperedges is correct up to $\mathcal{O}_d(\log^{2d-3} n)$ factor. Lemma 4.1 and 4.2 are intermediate results needed to prove Lemma 3.1; they bound the probability from above and below, respectively of Verify-Estimate accepting the estimate $\widehat{\mathcal{R}}$.

▶ **Lemma 4.1.** *If $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d\ m_o(\mathcal{H}_o)\log^{2d-3} n$, then*

$$\mathbb{P}(\text{Verify-Estimate } (\mathcal{H}_o,\widehat{\mathcal{R}}) \text{ accepts the estimate } \widehat{R}) \leq \frac{1}{20\cdot 2^d}.$$

Algorithm 1 VERIFY-ESTIMATE $(\mathcal{H}_o, \widehat{\mathcal{R}})$.

```
Input: CID query access to a $d$-uniform hypergraph $\mathcal{H}_o(U, \mathcal{F})$ and a guess $\widehat{R}$ for the
       number of hyperedges in $\mathcal{H}_o$.
Output: ACCEPT $\widehat{\mathcal{R}}$ or REJECT $\widehat{\mathcal{R}}$.
Let
   $U_1 = \ldots = U_d = U(\mathcal{H})$ for $(j_1 = d \log n$ to $0)$ do
    find $B_1 \subseteq U_1$ by sampling every element of $U_1$ with probability $p_1 = \min\left\{\frac{2^{j_1}}{\widehat{\mathcal{R}}}, 1\right\}$
     independently of other elements.
    for $(j_2 = d \log n$ to $0)$ do
        find $B_2 \subseteq U_2$ by sampling every element of $U_2$ with probability
         $p_2 = \min\left\{2^{j_2 - j_1} \cdot d \log n, 1\right\}$ independently of other elements.
        ⋮
        for $(j_{d-1} = d \log n$ to $0)$ do
            find $B_{d-1} \subseteq U_{d-1}$ by sampling every element of $U_{d-1}$ with probability
             $p_{d-1} = \min\{2^{j_{d-1} - j_{d-2}} \cdot d \log n, 1\}$ independently of other elements.
            Let $\mathbf{j} = (j_1, \ldots, j_{d-1}) \in [(d \log n)^*]^{d-1}$
            Let $p(i, \mathbf{j}) = p_i$, where $1 \leq i \leq d-1$
            Let $B(i, \mathbf{j}) = B_i$, where $1 \leq i \leq d-1$
            find $B(d, \mathbf{j}) = B_d \subseteq U_d$ by sampling every element of $U_d$ with probability
             $p_d = \min\left\{2^{-j_{d-1}}, 1\right\}$ independently of other elements.
            if $(m_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}}) \neq 0)$ then
                ACCEPT /*[Note that CID$_2^o$ query is called in the above line.]*/
            end
        end
    end
end
REJECT
```

Proof. Consider the set of ordered hyperedges $\mathcal{F}_o(\mathcal{H}_o)$ in $\mathcal{H}_o$. Algorithm VERIFY-ESTIMATE taking parameters $\mathcal{H}_o$, and $\widehat{\mathcal{R}}$ and described in Algorithm 1, loops over all possible $\mathbf{j} = (j_1, \ldots, j_{d-1}) \in [(d \log n)^*]^{d-1}$ [5]. For each $\mathbf{j} = (j_1, \ldots, j_{d-1}) \in [(d \log n)^*]^{d-1}$, VERIFY-ESTIMATE $(\mathcal{H}_o, \widehat{\mathcal{R}})$ samples vertices in each U_i with *suitable* probability values $p(i, \mathbf{j})$, depending on $\mathbf{j}$, $\widehat{R}$, d and $\log n$, to generate the sets $B_{i,\mathbf{j}}$ for $1 \leq i \leq d$. See Algorithm 1 for the exact values of $p(i, \mathbf{j})$'s. VERIFY-ESTIMATE $(\mathcal{H}_o, \widehat{\mathcal{R}})$ reports ACCEPT if there exists one $\mathbf{j} \in [(d \log n)^*]^{d-1}$ such that $m_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}}) \neq 0$. Otherwise, REJECT is reported by VERIFY-ESTIMATE $(\mathcal{H}_o, \widehat{\mathcal{R}})$.

For an ordered hyperedge $F_o \in \mathcal{F}_o(\mathcal{H}_o) = \mathcal{F}_o(U_1, \ldots, U_d)$ and $\mathbf{j} \in [(d \log n)^*]^{d-1}$. Note that

$$U_1 = \ldots = U_d = U(\mathcal{H}).$$

Let $X_{F_o}^{\mathbf{j}}$ denote the indicator random variable such that $X_{F_o}^{\mathbf{j}} = 1$ if and only if $F_o \in \mathcal{F}_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}})$. Let

$$X_{\mathbf{j}} = \sum_{F_o \in \mathcal{F}_o(\mathcal{H}_o)} X_{F_o}^{\mathbf{j}}.$$

[5] Recall that $[n]^*$ denotes the set $\{0, \ldots, n\}$.

Note that $m_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}}) = X_{\mathbf{j}}$. We have,

$$\begin{aligned}\mathbb{P}\left(X_{F_o}^{\mathbf{j}} = 1\right) &= \prod_{i=1}^{d}(p(i,\mathbf{j})) \\ &\leq \frac{2^{j_1}}{\widehat{\mathcal{R}}} \cdot \frac{2^{j_2}}{2^{j_1}} d\log n \times \cdots \times \frac{2^{j_{d-1}}}{2^{j_{d-2}}} d\log n \times \frac{1}{2^{j_{d-1}}} \\ &= \frac{d^{d-2}\log^{d-2} n}{\widehat{\mathcal{R}}}\end{aligned}$$

Then,

$$\mathbb{E}\left[X_{\mathbf{j}}\right] \leq \frac{m_o(\mathcal{H}_o)}{\widehat{\mathcal{R}}} d^{d-2}\log^{d-2} n,$$

and since $X_{\mathbf{j}} \geq 0$, we have

$$\mathbb{P}\left(X_{\mathbf{j}} \neq 0\right) = \mathbb{P}(X_{\mathbf{j}} \geq 1) \leq \mathbb{E}\left[X_{\mathbf{j}}\right] \leq \frac{m_o(\mathcal{H}_o)}{\widehat{\mathcal{R}}} d^{d-2}\log^{d-2} n.$$

Now, using the fact that $\widehat{\mathcal{R}} \geq 20d^{2d-3} \cdot 4^d \cdot m_o(\mathcal{H}_o)\log^{2d-3} n$, we have

$$\mathbb{P}\left(X_{\mathbf{j}} \neq 0\right) \leq \frac{1}{20d^{d-1} \cdot 4^d \cdot \log^{d-1} n}.$$

Recall that Verify-Estimate accepts if and only if there exists $\mathbf{j}$ such that $X_{\mathbf{j}} \neq 0$ [6]. Using the union bound, we get

$$\begin{aligned}\mathbb{P}\left(\text{Verify-Estimate } (\mathcal{H}_o, \widehat{\mathcal{R}}) \text{ accepts the estimate } \widehat{R}\right) &\leq \sum_{\mathbf{j}\in[(d\log n)^*]^{d-1}} \mathbb{P}(X_{\mathbf{j}} \neq 0) \\ &\leq \frac{(d\log n + 1)^{d-1}}{20 \cdot 4^d \cdot (d\log n)^{d-1}} \\ &\leq \frac{1}{20 \cdot 2^d}.\end{aligned}$$ ◀

▶ **Lemma 4.2.** *If $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d\log n}$, $\mathbb{P}($*Verify-Estimate *$(\mathcal{H}_o, \widehat{\mathcal{R}})$ accepts the estimate $\widehat{R}) \geq \frac{1}{2^d}$.*

Proof. We will be done by showing the following. Verify-Estimate accepts with probability at least $1/5$ when the loop variables $j_1, \ldots, j_{d-1}$ respectively attain values $q_1, \ldots, q_{d-1}$ such that

$$|U_1(q_1)| > \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d\log n + 1)}$$

and

$$|U_i((Q_i, A_i), q_i)| > \frac{2^{q_{i-1}}}{2^{q_i+1}(d\log n + 1)}$$

for all $i \in [d-1] \setminus \{1\}$. The existence of such j_is is evident from Claim 3.2. Let $\mathbf{q} = (q_1, \ldots, q_{d-1})$. Recall that $B_{i,\mathbf{q}} \subseteq U_i$ is the sample obtained when the loop variables $j_1, \ldots, j_{d-1}$ attain values $q_1, \ldots, q_{d-1}$, respectively. Let $\mathcal{E}_i, i \in [d-1]$, be the events defined as follows.

[6] Note that $\mathbf{j}$ is a vector but $X_{\mathbf{j}}$ is a scalar.

- $\mathcal{E}_1 \; : \; U_1(q_1) \cap B_{1,\mathbf{q}} \neq \emptyset$.
- $\mathcal{E}_i \; : \; U_j((Q_{j-1}, A_{j-1}), q_j) \cap B_{j,\mathbf{q}} \neq \emptyset$, where $2 \leq i \leq d-1$.

As noted earlier, Claim 3.2 says that for each $i \in [d-1]$, there exists a bucket in U_i having a *large* number of vertices contributing approximately the same number of hyperedges. The above events correspond to the nonempty intersection of vertices in heavy buckets corresponding to U_i and the sampled vertices $B_{i,\mathbf{j}}$, where $i \in [d-1]$. Observe that

$$\begin{aligned}
\mathbb{P}(\overline{\mathcal{E}_1}) &\leq \left(1 - \frac{2^{q_1}}{\widehat{\mathcal{R}}}\right)^{|U_1(q_1)|} \\
&\leq \exp\left(-\frac{2^{q_1}}{\widehat{\mathcal{R}}}\,|U_1(q_1)|\right) \\
&\leq \exp\left(-\frac{2^{q_1}}{\widehat{\mathcal{R}}} \cdot \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d\log n + 1)}\right) \\
&\leq \exp(-1).
\end{aligned}$$

The last inequality uses the fact that $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d\log n}$, from the condition of the lemma. Assume that $\mathcal{E}_1$ occurs and $a_1 \in U_1(q_1) \cap B_{1,\mathbf{q}}$. We will bound the probability that $U_2(Q_1, A_1), q_2) \cap B_{2,\mathbf{q}} = \emptyset$, that is $\overline{\mathcal{E}_2}$. Note that, by Claim 3.2 (ii),

$$|U_2(Q_1, A_1), q_2)| \geq \frac{2^{q_1}}{2^{q_2+1}(d\log n + 1)}.$$

So,

$$\mathbb{P}\left(\overline{\mathcal{E}_2} \mid \mathcal{E}_1\right) \leq \left(1 - \frac{2^{q_2}}{2^{q_1}}\log n\right)^{|U_2(Q_1,A_1),q_2)|} \leq \exp(-1)$$

Assume that $\mathcal{E}_1, \ldots, \mathcal{E}_{i-1}$ hold, where $3 \leq i \in [d-1]$. Let $a_1 \in U_1(q_1)$ and $a_{i-1} \in A_{i-1}((Q_{i-2}, U_{i-2}), q_{i-1})$. We will bound the probability that $U_i((Q_{i-1}, A_{i-1}), q_i) \cap B_{i,\mathbf{q}} = \emptyset$, that is $\overline{\mathcal{E}_i}$. Note that

$$|U_i((Q_{i-1}, A_{i-1}), q_i)| \geq \frac{2^{q_{i-1}}}{2^{q_i+1}(d\log n + 1)}.$$

So, for $3 \leq i \in [d-1]$,

$$\mathbb{P}\left(\overline{\mathcal{E}_i} \mid \mathcal{E}_1, \ldots, \mathcal{E}_{i-1}\right) \leq \left(1 - \frac{2^{q_i}}{2^{q_{i-1}}}\log n\right)^{|U_i(Q_{i-1},A_{i-1}),q_i)|} \leq \exp(-1)$$

Assume that $\mathcal{E}_1, \ldots, \mathcal{E}_{d-1}$ hold. Let $a_1 \in U_1(q_1)$ and $a_{i-1} \in A_{i-1}((Q_{i-2}, A_{i-2}), q_{i-1})$ for all $i \in [d] \setminus \{1\}$. Let $S \subseteq U_d$ be the set of d-th vertex of the ordered hyperedges in $\mathcal{F}_o(\mathcal{H}_o)$ having u_j as the j-th vertex for all $j \in [d-1]$. Note that $|S| \geq 2^{q_{d-1}}$. Let $\mathcal{E}_d$ be the event that represents the fact $S \cap B_{d,\mathbf{q}} \neq \emptyset$. So,

$$\mathbb{P}(\overline{\mathcal{E}_d} \mid \mathcal{E}_1, \ldots, \mathcal{E}_{d-1}) \leq \left(1 - \frac{1}{2^{q_{d-1}}}\right)^{q_{d-1}} \leq \exp(-1)$$

Observe that Verify-Estimate accepts if $m(B_{1,\mathbf{q}}, \ldots, B_{d,\mathbf{q}}) \neq 0$. Also,

$$m_o(B_{1,\mathbf{q}}, \ldots, B_{d,\mathbf{q}}) \neq 0 \text{ if } \bigcap_{i=1}^{d} \mathcal{E}_i \text{ occurs.}$$

Hence,

$$\begin{aligned}\mathbb{P}(\textsc{Verify-Estimate}\ (\mathcal{H}_o, \widehat{\mathcal{R}})\text{ accepts}) &\geq \mathbb{P}\left(\bigcap_{i=1}^{d} \mathcal{E}_i\right)\\ &= \mathbb{P}(\mathcal{E}_1)\prod_{i=2}^{d} \mathbb{P}\left(\mathcal{E}_i \,\middle|\, \bigcap_{j=1}^{i-1}\mathcal{E}_j\right)\\ &> \left(1-\frac{1}{e}\right)^d\\ &> \frac{1}{2^d}.\end{aligned}$$

◀

Now, we will prove Lemma 3.1 that will be based on Algorithm 2.

Algorithm 2 Rough Estimation($\mathcal{H}_o(U, \mathcal{F}_o)$).

```
Input: CID$_2^o$ query access to a $d$-uniform hypergraph $\mathcal{H}_o(U, \mathcal{F}_o)$.
Output: An estimate $\widehat{m}_o$ for $m_o = m_o(\mathcal{H}_o)$.
for ( $\widehat{\mathcal{R}} = n^d, n^d/2, \ldots, 1$) do
    Repeat Verify-Estimate $(\mathcal{H}_o, \widehat{\mathcal{R}})$ for $\Gamma = d \cdot 4^d \cdot 2000 \log n$ times. If more than
    $\frac{\Gamma}{10 \cdot 2^d}$ Verify-Estimate accepts, then output $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d \cdot (\log n)^{d-2}}$.
end
```

Proof of Lemma 3.1. Note that an execution of Rough Estimation for a particular $\widehat{\mathcal{R}}$ repeats Verify-Estimate for $\Gamma = d \cdot 4^d \cdot 2000 \log n$ times and gives output $\widehat{\mathcal{R}}$ if more than $\frac{\Gamma}{10 \cdot 2^d}$ Verify-Estimate accepts. For a particular $\widehat{\mathcal{R}}$, let X_i be the indicator random variable such that $X_i = 1$ if and only if the i-th execution of Verify-Estimate accepts. Also take $X = \sum_{i=1}^{\Gamma} X_i$. Rough Estimation gives output $\widehat{\mathcal{R}}$ if $X > \frac{\Gamma}{10 \cdot 2^d}$.

Consider the execution of Rough Estimation for a particular $\widehat{\mathcal{R}}$. If $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d \cdot m_o(\mathcal{H}_o) \cdot \log^{2d-3} n$, then we first show that Rough Estimation does not accept with high probability. Recall Lemma 4.1. If $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d \cdot m_o(\mathcal{H}_o) \log^{2d-3} n$, $\mathbb{P}(X_i = 1) \leq \frac{1}{20 \cdot 2^d}$ and hence $\mathbb{E}[X] \leq \frac{\Gamma}{20 \cdot 2^d}$. By using Chernoff-Hoeffding's inequality (See Lemma A.2 (i) in Section A),

$$\mathbb{P}\left(X > \frac{\Gamma}{10 \cdot 2^d}\right) = \mathbb{P}\left(X > \frac{\Gamma}{20 \cdot 2^d} + \frac{\Gamma}{20 \cdot 2^d}\right) \leq \frac{1}{n^{10d}}$$

Using the union bound for all $\widehat{\mathcal{R}}$, the probability that Rough Estimation outputs some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d}$ such that $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d \cdot m_o(\mathcal{H}_o) \log^{2d-3} n$, is at most $\frac{d \log n}{n^{10}}$. Now consider the instance when the for loop in the algorithm Rough Estimation executes for a $\widehat{\mathcal{R}}$ such that $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$. In this situation, $\mathbb{P}(X_i = 1) \geq \frac{1}{2^d}$. So, $\mathbb{E}[X] \geq \frac{\Gamma}{2^d}$. By using Chernoff-Hoeffding's inequality (See Lemma A.2 (ii) in Section A),

$$\mathbb{P}\left(X \leq \frac{\Gamma}{10 \cdot 2^d}\right) \leq \mathbb{P}\left(X < \frac{\Gamma}{2^d} - \frac{4}{5} \cdot \frac{\Gamma}{2^d}\right) \leq \frac{1}{n^{100d}}$$

By using the union bound for all $\widehat{\mathcal{R}}$, the probability that Rough Estimation outputs some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d}$ such that $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$, is at most $\frac{d \log n}{n^{100d}}$. Observe that, the probability that Rough Estimation outputs some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d}$ such that $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d m_o(\mathcal{H}_o) \log^{2d-3} n$ or $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$, is at most

$$\frac{d\log n}{n^{10d}} + \frac{d\log n}{n^{100d}} \leq \frac{1}{n^{8d}}.$$

Putting everything together, Rough Estimation gives some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2}\cdot 2^d \cdot (\log n)^{d-2}}$ as the output with probability at least $1 - \frac{1}{n^{8d}}$ satisfying

$$\frac{m_o(\mathcal{H}_o)}{8d^{d-1}2^d \log^{d-1} n} \leq \widehat{m}_o \leq 20d^{d-1}2^d \cdot m_o(\mathcal{H}_o) \log^{d-1} n$$

From the pseudocode of Verify-Estimate (Algorithm 1), we call for CID$_2$ queries only at line number 12. In the worst case, Verify-Estimate executes line number 12 for each $\mathbf{j} \in [(d\log n)^*]$. That is, the query complexity of Verify-Estimate is $\mathcal{O}(\log^{d-1} n)$. From the description of Rough Estimation, Rough Estimation calls Verify-Estimate $\mathcal{O}_d(\log n)$ times for each choice of $\widehat{R}$. Hence, Rough Estimation makes $\mathcal{O}_d(\log^{d+1} n)$ CID$_2^o$ queries. ◀

5 Proof of Theorem 1.4

Before getting into the reasons *why Theorem 1.4 follows from Theorem 1.3*, let us first review the algorithms for d-Hyperedge-Estimation and d-Hyperedge-Sample by Dell et al. [9].

Overview of Dell et al. [9]

Dell et al.'s algorithm for d-Hyperedge-Sample make repeated calls to d-Hyperedge-Estimation. Their algorithm for d-Hyperedge-Estimation calls mainly three subroutines over $\mathcal{O}_d(\log n)$ iterations: Coarse, Halving, and Trim. Halving and Trim calls Coarse repeatedly. So, Coarse is the main building block for their algorithms for d-Hyperedge-Estimation and d-Hyperedge-Sample.

Coarse **algorithm**

It estimates the number of hyperedges in the hypergraph up to polylog factors by using polylog queries. The result is formally stated as follows, see [9, Sec. 4].

▶ **Lemma 5.1** (Coarse Algorithm by Dell et al. [9]). *There exists an algorithm* Coarse*, that has* CID *query access to a hypergraph $\mathcal{H}(U, \mathcal{F})$, makes $\mathcal{O}_d\left(\log^{2d+3} n\right)$* CID *queries, and finds $\widehat{m}$ satisfying*

$$\Omega_d\left(\frac{1}{\log^d n}\right) \leq \frac{\widehat{m}}{m} \leq \mathcal{O}_d\left(\log^d n\right)$$

with probability at least $1 - 1/n^{\Omega(d)}$.

▶ Remark. The objective of Coarse algorithm by Dell et al. is essentially same as that our Rough Estimation algorithm. Both of them can estimate the number of hyperedges in any induced subhypergrah. However, note that Rough Estimation (as stated in Theorem 1.3) has better approximation guarantee and better query complexity than that of Coarse algorithm of Dell et al. (as stated in Lemma 5.1).

The framework of Dell et al. implies that the query complexity of d-Hyperedge-Estimation and d-Hyperedge-Sample can be expressed by the approximation guarantee and the query complexity of the Coarse algorithm. This is formally stated as follows:

▶ **Lemma 5.2** (d-Hyperedge-Estimation and d-Hyperedge-Sample in terms of quality of Coarse algorithm [9]). *Let there exists an algorithm* Coarse*, that has* CID *query access to a hypergraph $\mathcal{H}(U,\mathcal{F})$, makes q* CID *queries, and finds $\widehat{m}$ satisfying $\frac{1}{b} \leq \frac{\widehat{m}}{m} \leq b$ with probability at least $1 - 1/n^{\Omega(d)}$. Then*

(i) d-Hyperedge-Estimation *can be solved by using*

$$\mathcal{O}_d\left(\log^2 n\left(\log nb + \frac{b^2\log^2 n}{\varepsilon^2}\right)q\right)$$

CID *queries.*

(ii) d-Hyperedge-Sample *can be solved by using*

$$\mathcal{O}_d\left(\log^6 n\left(\log nb + \frac{b^2\log^2 n}{\varepsilon^2}\right)q\right)$$

CID *queries.*

Why Theorem 1.4 follows from Theorem 1.3?

Observe that we get Proposition 1.2 (the result of Dell et al.) from Lemma 5.1 by substituting $b = \mathcal{O}_d\left(\log^d n\right)$ and $q = \mathcal{O}_d\left(\log^{2d+3} n\right)$ in Lemma 5.2. In Theorem 1.4 we improve on the Proposition 1.2 by using our main result (Theorem 1.3), and substituting $b = \mathcal{O}_d\left(\log^{d-1} n\right)$ and $q = \mathcal{O}_d\left(\log^{d+2} n\right)$ in Lemma 5.2.

The main reason we get an improved query complexity for hyperedge estimation in Theorem 1.4 as compared to Dell et al. (Proposition 5.2) is our Rough Estimation algorithm is an improvement over the Coarse algorithm of Dell et al. [9] in terms of approximation guarantee as well as query complexity.

How our Rough Estimation improves over Coarse of Dell et al. [9]?

At a very high level, the frameworks of our Rough Estimation algorithm and that of Dell et al.'s Coarse algorithm might look similar, but the main ideas involved are different. Our Rough Estimation (as stated in Lemma 3.1) directly deals with the hypergraph (though the ordered one) and makes use of CID$_2^o$ queries. Note that each CID$_2^o$ query can be simulated by using $\mathcal{O}_d(\log n)$ CID queries. However, Coarse algorithm of Dell et al. considers $\mathcal{O}_d(\log n)$ independent random d-partite hypergraphs by partitioning the vertex set into d parts uniformly at random, works on the d-partite hypergraphs, and reports the median, of the $\mathcal{O}_d(\log n)$ outputs corresponding to random d-partite subhypergrahs, as the final output. So, there is $\mathcal{O}_d(\log n)$ blowup in both our Rough Estimation algorithm and Dell et al.'s Coarse algorithm, though the reasons behind the blowups are different.

Our Rough Estimation calls repeatedly ($\mathcal{O}_d(\log n)$ times) Verify Estimate for each guess, where the total number of guesses is $\mathcal{O}_d(\log n)$. In the Coarse algorithm, Dell et al. uses repeated calls $\left(\mathcal{O}_d\left(\log^{d+1} n\right)\right)$ times to an analogous routine of our Verify Estimate, which they name Verify Guess, $\mathcal{O}_d(\log n)$ times. Their Verify Guess has the following criteria for any guess M:

- If $M \geq \frac{d^d \log^{2d} n}{2^{3d-1}} m$, Verify Guess accepts M with probability at most p;
- If $M \leq m$, Verify Guess accepts M with probability at least $2p$;
- It makes $\mathcal{O}_d\left(\log^d n\right)$ CID queries.

Recall that the number of CID_2 queries made by each call to Verify Estimate is $\mathcal{O}_d(\log^{d-1} n)$, that is, $\mathcal{O}_d\left(\log^d n\right)$ CID queries. So, in terms of the number of CID queries, both our Rough Estimation and Coarse of Dell et al. have the same complexity.

The probability p in Verify Guess of Dell et al. [9] satisfies $p \approx_d \frac{1}{\log^d n}$, where $\approx_d$ is used suppress the terms involving d. So, for each guess M , their Coarse algorithm has to call $\mathcal{O}_d\left(\frac{1}{p}\log n\right) = \mathcal{O}_d\left(\log^{d+1} n\right)$ times to distinguish decide whether it is the case $M \leq m$ or $M \geq \frac{d^d \log^{2d} n}{2^{3d-1}} m$, with a probability at least $1 - 1/n^{\Omega(d)}$. So, the total number of queries made by the Coarse algorithm of Dell et al. [9] is

$$\mathcal{O}_d(\log n) \cdot \mathcal{O}_d(\log n) \cdot \mathcal{O}_d\left(\log^{d+1} n\right) \cdot \mathcal{O}_d\left(\log^d n\right) = \mathcal{O}_d\left(\log^{2d+3} n\right).$$

The first $\mathcal{O}_d(\log n)$ term is due to the blow up incurred to convert original hypergraph to d-partite hypergraph, the second $\mathcal{O}_d(\log n)$ term is due to the number of guesses for m, the third $\mathcal{O}_d\left(\log^{d+1} n\right)$ term is the number of times Coarse calls Verify Guess, and the last term $\mathcal{O}_d\left(\log^d n\right)$ is the number of CID queries made by each call to Verify Guess.

As it can be observed from Lemmas 4.1 and 4.2, p in our case (Verify Estimate) is $\Omega_d(1)$. So, it is enough for Rough Estimation to call Verify Estimate only $\mathcal{O}_d(\log n)$ times. Therefore, the number of CID queries made by our Rough Estimation is

$$\mathcal{O}_d(\log n) \cdot \mathcal{O}_d(\log n) \cdot \mathcal{O}_d(\log^{d-1} n) \cdot \mathcal{O}_d(\log n) = \mathcal{O}_d(\log^{d+2} n).$$

In the above expression, the first $\mathcal{O}_d(\log n)$ term is due to the number of guesses for m, the second $\mathcal{O}_d\left(\log n\right)$ term is the number of times Rough Estimation calls Verify Estimate, the third $\mathcal{O}\left(\log^{d-1} n\right)$ term is the number of CID_2 queries made by each call to Verify Estimate, and the last $\mathcal{O}_d(\log n)$ term is the number of CID queries needed to simulate a CID_2 query with probability at least $1 - 1/n^{\Omega(d)}$.

We do the improvement in approximation guarantee as well as query complexity in Rough Estimation algorithm (as stated in Theorem 1.3), as compared to Coarse algorithm of Dell et al. [9] (as stated in Lemma 5.1), by a careful analysis of the intersection pattern of the hypergraphs and setting the sampling probability parameters in Verify Estimate (Algorithm 1) algorithm in a nontrivial way, which is evident from the description of Algorithm 1 and its analysis.

References

1 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, volume 94, pages 38:1–38:21, 2018.

2 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. *ACM Trans. Algorithms*, 16(4):52:1–52:27, 2020.

3 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle Estimation Using Tripartite Independent Set Queries. In *Proceedings of the 30th International Symposium on Algorithms and Computation, ISAAC*, volume 149, pages 19:1–19:17, 2019.

4 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. On Triangle Estimation Using Tripartite Independent Set Queries. *Theory Comput. Syst.*, 65(8):1165–1192, 2021.

5 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Faster Algorithms for Estimating and Sampling using Colorful Decision Oracle, 2022. `arXiv:2201.04975`.

6 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Parameterized Query Complexity of Hitting Set Using Stability of Sunflowers. In *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC*, volume 123, pages 25:1–25:12, 2018.

7 Holger Dell and John Lapinskas. Fine-Grained Reductions from Approximate Counting to Decision. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 281–288, 2018.

8 Holger Dell and John Lapinskas. Fine-Grained Reductions from Approximate Counting to Decision. *ACM Trans. Comput. Theory*, 13(2):8:1–8:24, 2021.

9 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2201–2211, 2020.

10 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, 2009.

11 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017. doi:10.1137/15M1054389.

12 Talya Eden, Dana Ron, and C. Seshadhri. On Approximating the Number of k-Cliques in Sublinear Time. *SIAM J. Comput.*, 49(4):747–771, 2020.

13 Uriel Feige. On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph. *SIAM J. Comput.*, 35(4):964–984, 2006.

14 Oded Goldreich and Dana Ron. Approximating Average Parameters of Graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

A Some probability results

▶ **Lemma A.1** (Chernoff-Hoeffding bound [10]). *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$, the followings hold for any $0 \leq \delta \leq 1$.*

$$\mathbb{P}(|X - \mu| \geq \delta\mu) \leq 2\exp\left(-\mu\delta^2/3\right)$$

▶ **Lemma A.2** (Chernoff-Hoeffding bound [10]). *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum_{i=1}^{n} X_i$ and $\mu_l \leq \mathbb{E}[X] \leq \mu_h$, the followings hold for any $\delta > 0$.*

(i) $\mathbb{P}\left(X > \mu_h + \delta\right) \leq \exp\left(-2\delta^2/n\right)$.

(ii) $\mathbb{P}\left(X < \mu_l - \delta\right) \leq \exp\left(-2\delta^2/n\right)$.

B Oracle definitions

▶ **Definition B.1** (Independent set query (IS) [1]). *Given a subset A of the vertex set V of a graph $G(V,E)$,* IS *query answers whether A is an independent set.*

▶ **Definition B.2** (Bipartite independent set oracle (BIS) [1]). *Given two disjoint subsets A, B of the vertex set V of a graph $G(V,E)$,* BIS *query reports whether there exists an edge having endpoints in both A and B.*

▶ **Definition B.3** (Tripartite independent set oracle (TIS) [3]). *Given three disjoint subsets A, B, C of the vertex set V of a graph $G(V,E)$, the* TIS *oracle reports whether there exists a triangle having endpoints in A, B and C.*

▶ **Definition B.4** (Generalized d-partite independent set oracle (CID) [6]). *Given d pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ as input,* CID *query answers whether $m(A_1, \ldots, A_d) \neq 0$, where $m(A_1, \ldots, A_d)$ denotes the number of hyperedges in $\mathcal{H}$ having exactly one vertex in each A_i, $\forall i \in \{1, 2, \ldots, d\}$.*

▶ **Definition B.5** (CID_1 oracle). *Given s pairwise disjoint subsets of vertices $A_1, \ldots, A_s \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ and $a_1, \ldots, a_s \in [d]$ such that $\sum_{i=1}^{s} a_i = d$,* CID_1 *query on input $A_1^{[a_1]}, A_2^{[a_2]}, \cdots, A_s^{[a_s]}$ answers whether $m(A_1^{[a_1]}, \ldots, A_s^{[a_s]}) \neq 0$.*

▶ **Definition B.6** (CID_2 oracle). *Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$,* CID_2 *query on input $A_1, \ldots, A_d$ answers whether $m(A_1, \ldots, A_d) \neq 0$.*

▶ **Definition B.7** (CID_2^o oracle). *Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H}_o)$ of an ordered hypergraph $\mathcal{H}_o$,* CID_2^o *query on input $A_1, \ldots, A_d$ answers* Yes *if and only if $m_o(A_1, \ldots, A_d) \neq 0$.*

Probabilistic vs Deterministic Gamblers

Laurent Bienvenu
LaBRI, CNRS & Université de Bordeaux, France

Valentino Delle Rose
Dipartimento di Ingegneria Informatica e Scienze Matematiche, University of Siena, Italy

Tomasz Steifer
Institute of Fundamental Technological Research, Polish Academy of Sciences, Warszawa, Poland

Abstract

Can a probabilistic gambler get arbitrarily rich when all deterministic gamblers fail? We study this problem in the context of algorithmic randomness, introducing a new notion – almost everywhere computable randomness. A binary sequence X is a.e. computably random if there is no probabilistic computable strategy which is total and succeeds on X for positive measure of oracles. Using the fireworks technique we construct a sequence which is partial computably random but not a.e. computably random. We also prove the separation between a.e. computable randomness and partial computable randomness, which happens exactly in the uniformly almost everywhere dominating Turing degrees.

2012 ACM Subject Classification Theory of computation → Computability

Keywords and phrases Algorithmic randomness, Martingales, Probabilistic computation, Almost everywhere domination

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.11

Funding *Tomasz Steifer*: Supported by the National Science Centre Poland grant no. 2018/31/B/HS1/04018 and the 2021 French Government Scholarship.

1 Introduction

What does it mean for an infinite binary sequence X to be random? This may seem like a strange question at first since in classical probability theory, any infinite binary sequence drawn at random (with respect to the uniform distribution) has probability 0 to occur. Yet, the theory of algorithmic randomness gives us a way answer it from a computability perspective: X is random if it does not possess any property of measure 0 which can be computably tested. There are many ways to formalize this, and hence many possible definitions of random sequence. One of the main approaches is the so-called unpredictability paradigm. We may say that a sequence X is unpredictable if no computable gambling strategy (or martingale) betting on the values of the bits of X and being rewarded fairly for its predictions can become arbitrarily rich during the course of the (infinite) game. The main two notions of randomness derived from this point of view are computable randomness and partial computable randomness, depending on whether we allow total computable or partial computable martingales. But in either case, the martingales considered are deterministic.

In this paper, we ask: do we get a stronger notion of randomness if we ask that X defeats not just all deterministically computable martingales, but also all probabilistically computable martingales? Usually, in computability theory, allowing probabilistic computations does not make a difference. This is in large part due to the foundational result that if a set $A \subset \mathbb{N}$ (or function $f : \mathbb{N} \to \mathbb{N}$, etc.) can be obtained by a probabilistic computation with positive probability, then it can in fact be obtained via a deterministic computation [5]. Yet this result is not necessarily an obstacle here as for a given X, different runs of the probabilistic algorithm are allowed to produce different martingales, as long as with positive probability,

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 11; pp. 11:1–11:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the martingale output by the probabilistic algorithm defeats X. And indeed, the main result of our paper is that probabilistic martingales *do* in fact perform better than deterministic ones!

We should note that probabilistic martingales were already considered by Buss and Minnes [4]. However, the applicability of their results for our purpose is limited. In particular, they studied two cases: probabilistic martingales which are total almost surely and probabilistic martingales which may be partial but nevertheless almost surely succeed on a given sequence. It is fairly easy to show that these cases reduce to computable and partial computable martingales respectively. The results of this paper are different and require more involved proofs.

1.1 Notation

The set of all infinite binary sequences is denoted by $2^{\mathbb{N}}$, while the set of finite binary strings is $2^{<\mathbb{N}}$. The truncation of x to the first n bits is $x \upharpoonright n$, while length of a string σ is written by $|\sigma|$. We write $\tau \prec x$ when τ is a prefix of some x (which might be a sequence or a string). The empty string is denoted by ϵ, the concatenation of two strings σ and τ by $\sigma^\frown\tau$. We are working with the product topology on $2^{\mathbb{N}}$, i.e., the topology generated by cylinder sets $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \prec X\}$. This means that open sets are of the form $\bigcup_{\sigma\in A}[\sigma]$ where A is any set of strings. When A is computably enumerable (c.e.), the set $\bigcup_{\sigma\in A}[\sigma]$ is called *effectively open*. In this topology, the clopen sets are exactly the finite unions of cylinders.

We further equip $2^{\mathbb{N}}$ with the uniform measure μ, which is the measure where each bit of the sequence is equal to $1/2$ independently of the values of other bits. Formally, μ is the unique probability measure on the σ-algebra generated by cylinders for which $\mu([\sigma]) = 2^{-|\sigma|}$ for all σ.

As is common in computability theory, we sometimes identify sequences and strings with subsets of $\mathbb{N}$ (via characteristic function of the set) or paths in the full infinite binary tree. In particular, we say that σ is on the left of τ if σ is lesser than τ with respect to the lexicographical order.

1.2 Algorithmic randomness

Algorithmic randomness' goal is to assign a meaning to the notion of individual random string or sequence. While for strings we cannot reasonably hope for a clear separation between random and non-random (instead we have a quantitative measure of randomness: Kolmogorov complexity), for infinite binary sequences one can get such a separation. There are in fact many possible definitions. The most important one is called Martin-Löf randomness and is defined as follows. A set $\mathcal{N} \subset 2^{\mathbb{N}}$ is called effectively null if for every n one can cover it by an effectively open set of measure at most $\leq 2^{-n}$, uniformly in n.

▶ **Definition 1.** *A sequence $X \in 2^{\mathbb{N}}$ is called Martin-Löf random if it does not belong to any effectively null set.*

Said otherwise, X is Martin-Löf random if for every sequence $(\mathcal{U}_n)$ of uniformly effectively open sets such that $\mu(\mathcal{U}_n) \leq 2^{-n}$ for all n (such a sequence is known as a *Martin-Löf test*), we have $X \notin \bigcap_n \mathcal{U}_n$.

An effectively null set corresponds to an atypical (= measure 0) property which can in some sense be effectively tested and therefore, a Martin-Löf random sequence is one that withstands all computable statistical tests. The reason Martin-Löf's definition of randomness is considered to be the central one is that it is both well-behaved (Martin-Löf random sequences possess most properties one would expect from "random" sequences, including computability-theoretic properties) and robust, in that one can naturally get to the same notion by different approaches. For example, if we denote by K the prefix-free Kolmogorov complexity function (see for example [11]), then the Levin-Schnorr theorem states that a sequence X is Martin-Löf random if and only if $K(X \upharpoonright n) \geq n - d$ for some d and all n. Informally, this means that Martin-Löf random sequences are exactly the "incompressible" ones.

As discussed above there is, however, another natural paradigm to define randomness (seemingly different from atypicality): unpredictability. We want to say that a sequence X is random if its bits cannot be guessed with better-than-average accuracy. This is formalized via the notion of martingale.

▶ **Definition 2** (martingale). *A function $d : 2^{<\mathbb{N}} \to \mathbb{R}^{>0}$ is called a martingale if for all $\sigma \in 2^{<\mathbb{N}}$:*

$$d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}.$$

A martingale d succeeds on a sequence X if

$$\limsup_{n\to\infty} d(X \upharpoonright n) = \infty.$$

A martingale represents the outcome of a gambling strategy in a fair game where the gambler guesses bits one by one by betting some amount of money at each stage, doubling the stake if correct, losing the stake otherwise, debts not being allowed. The quantity $d(\sigma)$ represents the capital of the gambler after having seen σ. Usually in the literature martingales are allowed to take value 0 but not allowing it makes no difference for the definitions that follow and avoids some pathological cases later in the paper.

Armed with the notion of martingale, we can now formulate an important definition of "randomness", known as computable randomness.

▶ **Definition 3.** *A sequence $X \in 2^{\mathbb{N}}$ is called computably random if no computable martingale succeeds on X.*

In the above definition, we consider only martingales that are total computable. We would also like to allow partial computable martingales, but since they are not total functions in general, they are not even martingales in the above sense. To remedy this, one can simply define a partial martingale as a function d taking values in $\mathbb{R}^{>0}$ whose domain is contained in $2^{<\mathbb{N}}$ and closed under the prefix relation (if $d(\sigma)$ is defined, $d(\tau)$ is defined for every prefix τ of σ) and furthermore for every σ, $d(\sigma 0)$ is defined if and only if $d(\sigma 1)$ is defined and in case both are defined, the fairness condition $d(\sigma) = (d(\sigma 0) + d(\sigma 1))/2$ applies. Finally, success is defined in the same way as for martingales: we say that d succeeds on X if $d(X \upharpoonright n)$ is defined for all n and $\limsup_{n\to\infty} d(X \upharpoonright n) = \infty$. We can now get the following strengthening of computable randomness.

▶ **Definition 4.** *A sequence $X \in 2^{\mathbb{N}}$ is called partial computably random if no partial computable martingale succeeds on X.*

It is well-known that partial computable randomness is strictly stronger than computable randomness, but nonetheless strictly weaker than Martin-Löf randomness (see [11]).

Computable randomness and partial computable randomness are pretty robust notions. For example, it makes no difference whether we define success as achieving unbounded capital or as having a capital that tends to infinity.

▶ **Lemma 5** (folklore, see [7]). *For every total (resp. partial) computable martingale d there exists a (resp. partial) computable martingale d' such that d and d' succeed on exactly the same sequences and for every $A \in 2^{\mathbb{N}}$ we have $\limsup_{n\to\infty} d(A \upharpoonright n) = \infty$ iff $\lim_{n\to\infty} d'(A \upharpoonright n) = \infty$. Moreover, an index for d' can be found effectively from an index for d.*

Another important fact is that instead of considering computable real-valued martingales, we can restrict ourselves to rational valued martingales that are computable as functions from $2^{<\mathbb{N}}$ to $\mathbb{Q}$ (which we sometimes refer to as *exactly computable* martingales).

▶ **Lemma 6** (Exact Computation lemma, see [14]). *For every total (resp. partial) computable martingale d, there exists a total (resp. partial) exactly computable martingale d' such that d' succeeds on every sequence on which d succeeds. Moreover, an index for d' can be effectively obtained from an index for d.*

1.3 Probabilistic martingales

The above definitions assume computable martingales (partial or total) are deterministic. Our goal is to understand whether probabilistic martingales (i.e., obtained by a probabilistic algorithm) can do better. Usually, to capture the idea of probabilistic algorithm, one appeals to probabilistic models of computation, such as probabilistic Turing machines. However, from a computability-theoretic perspective, where relativization to an oracle is a bread-and-butter object of study, it is equivalent to assume that an infinite sequence of random bits is drawn in advance and given as oracle to a deterministic Turing machine which then uses it as a source of randomness. Thus, we will consider *partial computable oracle martingales*, that is, Turing functionals d where for every oracle Y, d^Y (the function computed by the functional with Y given as oracle) is a partial martingale.

▶ **Definition 7.** *A sequence $X \in 2^{\mathbb{N}}$ is called a.e. computably random if for every partial computable oracle martingale d the set of oracles Y such that d^Y is a total martingale and succeeds on X has measure zero, i.e.*

$$\mu\left(\left\{Y \in 2^{\mathbb{N}} : d^Y \text{ is total and } \limsup_{n\to\infty} d^Y(X \upharpoonright n) = \infty\right\}\right) = 0.$$

X is said to be a.e. partial computably random if for every partial computable oracle martingale d the set of oracles Y such that d^Y succeeds on X has measure zero.

Note that we could have equivalently defined a.e. (partial) computably randomness directly from the relativization of (partial) computable randomness: a sequence X is a.e. (partial) computably random if for almost every Y, X is (partial) computably random relative to Y.

The informal question "do probabilistic gamblers perform better than deterministic ones" can now be fully formalized by the following two questions:

- Is a.e. computable randomness equal to computable randomness?
- Is a.e. partial computable randomness equal to partial computable randomness?

In [4], Buss and Minnes studied a restricted version of this problem. They considered a model of probabilistic martingales where one further requires $d^Y(\sigma)$ to be defined for all σ and almost all Y. This is a strong restriction which allows one to use an averaging technique. If d is a probabilistic martingale with this property, it is easy to prove that the average D defined by $D(\sigma) = \int_Y d^Y(\sigma)$ is a computable martingale. If X is computably random, D fails against X, that is, there is a constant c such that $D(X \upharpoonright n) < c$ for all n. Moreover, by Fatou's lemma:

$$\int_Y \liminf_n d^Y(X \upharpoonright n) \leq \liminf_n D(X \upharpoonright n) < c \qquad (\star)$$

which in turn implies that the set $\{Y : \liminf_n d^Y(X \upharpoonright n) = \infty\}$ has measure 0. In other words, the set of Y such that d^Y strongly succeeds against X has measure 0. By Lemma 5, this means that if a sequence X is computably random if and only if for every probabilistic martingale with the Buss-Minnes condition, d fails on X with probability 1.

Our main result is that, in the general case, we no longer have an equivalence of the two models: probabilistic martingales are indeed stronger than deterministic ones.

▶ **Theorem 8.** *There exist a sequence X which is partial computably random but not a.e. partial computably random and indeed not even a.e. computably random.*

We will devote the next sections to proving Theorem 8, but let us say a few words on why we believe it to be an interesting result. First of all, it is in stark contrast with Buss and Minnes' result that probabilistic martingales do not do any better than deterministic ones when they are required to be total with probability 1: in the general case, probabilistic martingales do better! Second, this is to our knowledge the first result of this kind in algorithmic randomness. If we were to define a.e. Martin-Löf randomness following the same idea (i.e., saying that X is a.e. Martin-Löf random if for almost all Y, X is Martin-Löf random relative to oracle Y), we would not get anything new, because a.e. Martin-Löf randomness coincides with Martin-Löf randomness. This is a direct consequence of the famous van Lambalgen theorem [15], which states that for every $A, B \in 2^{\mathbb{N}}$, the join $A \oplus B = A(0)B(0)A(1)B(1)\dots$ is Martin-Löf random if and only if A is Martin-Löf random and B is Martin-Löf relative to A, if and only if B is Martin-Löf random and A is Martin-Löf random relative to B. Now, let X be Martin-Löf random. For almost all Y, Y is Martin-Löf random relative to X (this is simply the fact that the set of Martin-Löf random sequences has measure 1, relativized to X), thus $X \oplus Y$ is Martin-Löf random, and thus X is Martin-Löf random relative to Y. This shows that X is a.e. Martin-Löf random. We see that van Lambalgen's theorem is key in this argument (we use it three times!). It was already known that the analogue of van Lambalgen for computable randomness fails [16], but Theorem 8 shows that it fails in a very strong sense.

Let us also remark that van Lambalgen's theorem shows that Martin-Löf randomness implies a.e. (partial) computable randomness: if X is Martin-Löf random, it is also Martin-Löf random relative to Y for almost every Y, and thus also (partial) computably random relative to Y for almost every Y.

2 Turing degrees of a.e.CR sequences

Before moving to the proof of Theorem 8, we give a simple degree-theoretic proof of a weaker result, namely a separation between computable randomness and a.e. computable randomness.

Recall that every Martin-Löf random sequence is computably random but a computable random sequence is not necessarily Martin-Löf random.This separation has some interesting connections with classical computability theory, as witnessed by the following theorem (recall that a sequence Y has *high Turing degree*, or simply *is high* if it computes some function $F : \mathbb{N} \to \mathbb{N}$ such that for every total computable function f, $f(n) \leq F(n)$ for almost all n).

▶ **Theorem 9** (Nies, Stephan, Terwijn [12]). *Let $Y \in 2^{\mathbb{N}}$. If Y computes a sequence X such that X is computably random but not Martin-Löf random, then Y has high Turing degree. Conversely, if Y has high Turing degree, then it computes some X which is computably random but not Martin-Löf random.*

It turns out that one can get an exact analogue of this theorem for a.e. computable randomness by replacing highness with a stronger notion: almost everywhere domination. A sequence Y is said to have *almost everywhere dominating Turing degree, or a.e. dominating Turing degree* if it computes an almost everywhere dominating function F, that is, a function F such that for every Turing functional Γ and almost every Z, if Γ^Z is total, then $\Gamma^Z(n) \leq F(n)$ for almost all n. See [11] for a more complete presentation of the history of this notion, originally due to Dobrinen and Simpson [6].

▶ **Theorem 10.** *Let $Y \in 2^{\mathbb{N}}$. If Y computes a sequence X such that X is a.e. computably random but not Martin-Löf random, then Y has a.e. dominating Turing degree. Conversely, if Y has a.e. dominating Turing degree, then it computes some X which is a.e. computably random but not Martin-Löf random (in fact, it even computes some X which is a.e. computably random but not partial computably random).*

▶ Remark 11. Nies et al.'s theorem actually states a little more than what we wrote above, namely that the sequence X in the second part of the theorem can be chosen to be Turing equivalent to Y. The analogue theorem is also true for a.e. computable randomness and a.e. domination but the proof becomes substantially more technical (we would need to introduce techniques to encode information into a computably random sequence) for only a small gain.

Proof. Let us prove the first part of the theorem by its contrapositive. Let $X \in 2^{\mathbb{N}}$ whose degree is not almost everywhere dominating. Suppose also X is not Martin-Löf random, i.e., $X \in \bigcap_n \mathcal{U}_n$ for $(\mathcal{U}_n)_{n\in\mathbb{N}}$ a sequence of uniformly effectively open sets with $\mu(\mathcal{U}_n) \leq 2^{-n}$. Consider the function t^X defined by $t^X(n) := \min\{s \mid X \in \mathcal{U}_n[s]\}$. Since X does not have a.e. dominating degree, there must exist a functional Γ such that

$$\mu\{Z \mid \Gamma^Z \text{is total and } \exists^\infty n \; \Gamma^Z(n) > t^X(n)\} > 0.$$

When Γ^Z is total and $\Gamma^Z(n) > t^X(n)$ for infinitely many n, we have $X \in \mathcal{U}_n[\Gamma^Z(n)]$ for infinitely many n. Note that in that case $\mathcal{U}_n[\Gamma^Z(n)]$ is a clopen set which Z-uniformly computable in Z. It is well-known that this type of test characterizes Schnorr randomness (a notion we will no discuss here but suffices to say that Schnorr randomness is weaker than computable randomness): a sequence X is Schnorr random if and only if for every computable sequence of clopen sets $\mathcal{D}_n$ such that $\mu(\mathcal{D}_n) \leq 2^{-n}$, X belongs to only finitely $\mathcal{D}_n$ (see for example [1, Lemma 1.5.9]). Relativized to Z, this fact shows that X is not Z-Schnorr random for a positive measure of Z's, thus not Z-computably random for a positive measure of Z's.

The strategy to prove the second part of the theorem is to take the function F computed by Y and use it as a time bound on oracle martingales in order to "totalize" them, which then allows us to use the averaging argument presented on page 4. In order for this to work, we must first prove that F can be assumed to be "simple" (in terms of Kolmogorov complexity).

▶ **Lemma 12.** *If Y has a.e. dominating Turing degree, it computes an a.e. dominating function F such that $K(F(n)) = O(\log n)$.*

Proof. Let $(\Phi_i)_{i\in\mathbb{N}}$ be an enumeration of all Turing functionals and consider the universal functional Ψ where $\Psi^{0^i 1A} = \Phi_i^A$. It is easy to see that a function F is almost everywhere dominating if for almost all Z, either Ψ^Z is not total or $\Phi^Z(n) \leq F(n)$ for almost every n. For each Z, let $t^Z(n)$ be the minimum t, if it exists, such that $\Phi^Z(k)$ converges in time $\leq t$ for all $k \leq n$ and let $f^Z(n) = t^Z(n) + \max_{k\leq n} \Phi^Z(k)$.

Let Y be of a.e. dominating degree and $F \leq_T Y$ an almost everywhere dominating function.

For each n, let

$$\mathcal{U}_n = \{Z \mid f^Z(n) \downarrow < \infty\}$$

which is Σ^0_1 uniformly in n. We can write

$$\mathcal{U}_n = \bigcup_k \mathcal{U}_{n,k}$$

where

$$\mathcal{U}_{n,k} = \{Z \mid f^Z(n) \downarrow < k\}$$

and note that $\mathcal{U}_{n,k}$ is a clopen set, computable uniformly in n, k.

Since F is almost everywhere dominating, we have that for almost all Z and almost all n, either $f^Z(n)$ is undefined or $f^Z(n) \leq F(n)$. Said otherwise, the set

$$\mathcal{N}_0 = \limsup(\mathcal{U}_n \setminus \mathcal{U}_{n,F(n)})$$

is a nullset.

Now, for all n, let $a_n \in [0, n^2]$ be the largest integer that $\mu(\mathcal{U}_{n,F(n)}) \geq a_n/n^2$ and $F'(n)$ be the smallest k such that $\mu(\mathcal{U}_{n,k}) \geq a_n/n^2$. We see that $F'(n)$ is computable from F and furthermore,

$$K(F'(n)) \leq K(a_n) + O(1) \leq 2\log(n^2) + O(1) \leq 4\log n + O(1).$$

By definition, we have $\mu(\mathcal{U}_{n,F(n)}) \setminus \mathcal{U}_{n,F'(n)}) \leq 1/n^2$. By the Borel-Cantelli lemma,

$$\mathcal{N}_1 = \limsup(\mathcal{U}_{n,F(n)} \setminus \mathcal{U}_{n,F'(n)})$$

is a nullset. Thus, $\mathcal{N}_0 \cup \mathcal{N}_1$ is a nullset, which means that

$$\limsup(\mathcal{U}_n \setminus \mathcal{U}_{n,F'(n)})$$

is also a nullset, which in turn means that for almost all Z, for almost all n, if $f^Z(n)$ is defined, then $f^Z(n) \leq F'(n)$. By definition of f, a fortiori, for almost all Z, if Φ^Z is total, then $\Phi^Z(n) \leq F'(n)$ for almost all n. Thus the function F'

- is almost everywhere dominating
- is computable in F, hence computable in Y
- satisfies $K(F'(n)) = O(\log n)$

which finishes the proof of the lemma. ◀

As alluded to above, the function F is going to be used as a time bound. To see what we mean by this, consider a total (not necessarily computable) non-decreasing function $\psi : \mathbb{N} \to \mathbb{N}$. Let d be a (partial) exactly computable martingale. The time-bounded version of d with time bound ψ is the martingale d^ψ which mimics d but only allows it a time $\psi(n)$ to compute its bets on strings of length n. If d has not made a decision by this stage (either because it is in fact undefined, or because the time of computation is greater than $\psi(n)$)), the casino exclaims *"End of bets, nothing goes on the table!"* and the martingale is assumed to have placed an empty bet. Formally, $d^\psi(\epsilon) = d(\epsilon)$ and for any string σ and $b \in \{0, 1\}$:

$$d^\psi(\sigma b) = \begin{cases} d^\psi(\sigma) \cdot d(\sigma b)/d(\sigma) & \text{if both } d(\sigma 0)[\psi(n+1)] \downarrow \text{ and } d(\sigma 1)[\psi(n+1)] \downarrow \\ d^\psi(\sigma) \text{ otherwise.} \end{cases}$$

By definition d^ψ is always total, and when d is total, if the bound ψ dominates the convergence time of d (that is, for almost all σ, $d(\sigma)[\psi(|\sigma|)]\downarrow$), then d^ψ and d are within a multiplicative constant of one another, which in particular implies that d^ψ succeeds on the same sequences as d.

Now, let (d_i) be the effective enumeration of all (partial) exactly computable martingales with oracle. Without loss of generality, assume that d_i has a delay i imposed on it. Let F be the a.e dominating function as above. Let $\hat{d}$ be the oracle martingale defined by

$$\hat{d}^Z(\sigma) = \sum_i 2^{-i} d_i^{Z,F}(\sigma)$$

($d_i^{Z,F}$ is the time-bounded version of d_i^Z with time bound F).

It is a total martingale for every Z as all $d_i^{Z,F}$ are total martingales. Thus, its average D defined by

$$D(\sigma) = \int_Z \hat{d}^Z(\sigma)$$

is also a martingale.

Moreover, D is F- (exactly)computable. Indeed, because of the time bound F, the value of $d_i^{Z,F}(\sigma)$ only depends of the first $F(|\sigma|)$ bits of Z, and because of the delay on the d_i, only the martingales $(d_i)_{i\leq|\sigma|}$ matter in the computation of $D(\sigma)$. Thus the integral $\int_Z \hat{d}^Z(\sigma)$ is in fact a finite sum, can be computed from $F(|\sigma|)$, hence the F-computability of D. Even more precisely, the set of values $\{D(\sigma) \mid |\sigma| \leq n\}$ is computable from $F(n)$, and thus the Kolmogorov complexity of this set is at most $K(F(n)) + O(1) = O(\log n)$.

Let then X be the sequence which diagonalizes against D (that is, the sequence X constructed bit by bit where at each stage the chosen value of the next bit is the one that makes the martingale D lose money; all this will be detailed in the next section). Computing the first n bits of X only requires to know the set of values $\{D(\sigma) \mid |\sigma| \leq n\}$. Thus, we have established:

- $X \leq_T F$
- $K(X \upharpoonright n) \leq K(F(n)) + O(1) = O(\log n)$.

Since D does not succeed on X, by the exact same calculation as ($\star$) (see page 4), for almost all Z, $\hat{d}^Z$ does not succeed on X, and thus $d_i^{Z,F}$ does not succeed on X for any i.

But we also know, since F is a.e. dominating, for all i, for almost every Z, either d_i^Z is partial, or d_i^Z is total and its computation time is dominated by F, hence d^Z is within a multiplicative constant of $d^{Z,F}$.

Putting the two together, this entails that for almost all i and almost all Z, either d_i^Z is partial or it is total and does not succeed on X. In other words, X is a.e. computably random.

X has therefore all the desired properties:

- It is a.e. computably random,
- It is computable in F and thus computable in Y,
- $K(X \upharpoonright n) = O(\log n)$, ensuring that X is not only not Martin-Löf random, but not even partial computably random using a result of Merkle [10] (no partial computably random sequence can be of logarithmic complexity). ◀

An important result of Binns et al. [3] is that a.e. domination is strictly stronger than highness. This gives us the promised weaker version of Theorem 8.

▶ **Corollary 13.** *There exists a sequence X which is computably random but not a.e. computably random.*

Proof. Indeed, by Binn et al.'s result, take a high Turing degree $\mathbf{a}$ which is not a.e. dominating. By Theorem 9, there is an X in $\mathbf{a}$ which is computably random but not Martin-Löf random. By Theorem 10, X is not a.e. computably random either. ◀

3 The main construction

We now turn to the full proof of Theorem 8. We first recall the standard method to build a partial computably random sequence (see for example [11]). Next, we combine this construction with the so-called "fireworks" technique which can be viewed as a probabilistic forcing to see how to defeat, with probabilistic martingales, sequences that have been built using this construction.

3.1 Defeating finitely many martingales

Let us begin by explaining how to construct a partial computably random sequence. Let us first consider the simple case where we are trying to defeat a single martingale d, which we assume for the moment to be total computable, by making sure its capital does not go above a certain threshold. Up to multiplying d by a small rational, we may assume that that $d(\epsilon) < 1$. By induction, suppose we have already built $X \upharpoonright n$ in a way that $d(X \upharpoonright i) < 1$ for all $i \leq n$. By the fairness condition, either $d((X \upharpoonright n)^\frown 0) < 1$ or $d((X \upharpoonright n)^\frown 1) < 1$. If the former is true, we set $X \upharpoonright (n+1) = (X \upharpoonright n)^\frown 0$, otherwise we set $X \upharpoonright (n+1) = (X \upharpoonright n)^\frown 1$. Continuing in this fashion we ensure that the martingale d does not succeed against X as its never reaches 2. Observe that when the martingale d is exactly computable, the sequence X is computable (uniformly in a code for d).

Suppose now that we have a finite family of total martingales $d_1, \dots d_n$. If we want to diagonalize against all of them at the same time, one can simply find positive rationals $q_1, \dots, q_n$ such that $\sum_{i=1}^n q_i \cdot d_i(\epsilon) < 1$ and proceed as before against the martingale $\sum_{i=1}^n q_i \cdot d_i$. Again, the sequence X obtained by diagonalization against this finite family of martingales is computable uniformly in a code for the family of d_i's. But suppose now that some of the martingales in this family are partial instead of total. This does not cause much difficulty: having already built $X \upharpoonright n$, consider only the sub-family F of indices of martingales that are still defined on $(X \upharpoonright n)^\frown 0$ and $(X \upharpoonright n)^\frown 1$. The other martingales are undefined and thus will not succeed by fiat on the sequence X. Now, if $\sum_{i \in F} q_i \cdot d_i((X \upharpoonright n)^\frown 0) < 1$, set $X \upharpoonright (n+1) = (X \upharpoonright n)^\frown 0$, otherwise set $X \upharpoonright (n+1) = (X \upharpoonright n)^\frown 1$. Once again the sequence X defeats all of the d_i's, some of them because they become undefined at some stage, some of them because their capital never exceeds $1/q_i$. Moreover, X is still a computable sequence. It is not however computable uniformly in a code for the family of d_i's because one needs to specify which martingales become undefined in the construction and when (this is a finite amount of information but it cannot be uniformly computed) but this is not an obstacle for our purposes.

To summarize these preliminary considerations, we can make the following definition.

▶ **Definition 14.** *Let $(d_1, q_1), \dots (d_n, q_n)$ be a finite family where each d_i is a (code for) a partial computable martingale and q_i a positive rational. Let $\sigma \in 2^{<\mathbb{N}}$ such that, calling F the family of indices i such that $d_i(\sigma)$ converges, we have $\sum_{i \in F} q_i \cdot d_i(\sigma) < 1$. Consider the computable sequence X defined inductively by $X \upharpoonright |\sigma| = \sigma$ and if $X \upharpoonright n$ is already built, letting F_n be the family of indices such that $d_i((X \upharpoonright n)^\frown 0)$ converges, then $X \upharpoonright (n+1) = (X \upharpoonright n)^\frown 0$ if $\sum_{i \in F_n} q_i \cdot d_i(X \upharpoonright n)^\frown 0) < 1$ and $X \upharpoonright (n+1) = (X \upharpoonright n)^\frown 1$ otherwise. This sequence is called the* diagonalization against $(d_1, q_1), \dots, (d_n, q_n)$ above σ.

3.2 Defeating all partial computable martingales

When we have a countable family of martingales to diagonalize against, the standard way to proceed is to introduce them one by one during the game so that at any step we only have to diagonalize against a finite family as above. The delays between the introduction of martingales is flexible and therefore will be a parameter of the construction.

The diagonalizing sequence $\Delta((t_e)_{e\in N})$

Let $(d_i)_{i\in\mathbb{N}}$ be a standard enumeration of partial computable rational valued martingales. Let $(t_e)_{e\in\mathbb{N}}$ be a family of integers. The sequence $\Delta((t_e)_{e\in N})$ is constructed by finite extension as follows. Start with the empty string $\sigma_0 = \epsilon$ and recursively do the following. Having built σ_n, let q_{n+1} be a rational such that $\sum_{i\in F} q_i \cdot d_i(\sigma_n) < 1$ where F is the set of indices $i \in [1, n+1]$ such that $d_i(\sigma_n)$ converges. Let A be the diagonalization against $(d_1, q_1), \ldots, (d_{n+1}, q_{n+1})$ above σ_n. The sequence A is an extension of σ and is computable (see above), so let e be a code for it (say the smallest one). Define $\sigma_{n+1} = A \upharpoonright (|\sigma_n| + t_e)$. Finally, set

$$\Delta((t_e)_{e\in N}) = \bigcup_n \sigma_n.$$

It is easy to check that $\Delta((t_e)_{e\in N})$ defeats all partial computable martingales. Moreover, the construction ensures the following important fact, which will be key for the rest of our proof:

Fact 1: For infinitely many e (namely, those codes that show up in the construction), the sequence $\Delta((t_e)_{e\in N})$ coincides with the computable sequence A of index e on a prefix of length $\geq t_e$.

3.3 Fireworks

Let $(\mathbb{P}, \leq)$ be a computable order, that is, each element $p \in \mathbb{P}$ can be encoded by an integer and for a given pair (n, m) of integers, it is decidable whether n and m are indeed codes for two elements of p and q in $\mathbb{P}$ and whether $p \leq q$. We say that a sequence $(p_i)_{i\in\mathbb{N}}$ of elements of $\mathbb{P}$ is $\mathbb{P}$-*generic* if $p_0 \geq p_1 \geq p_2 \geq \ldots$ and for every c.e. subset W of $\mathbb{P}$:

- either there exists an i such that $p_i \in W$
- or, there exists a j such that for any $q \leq p_j$, $q \notin W$

In particular, if W is dense (that is, for every $p \in \mathbb{P}$ there exists $q \leq p$ such that $q \in W$), then for every generic sequence $(p_i)_{i\in\mathbb{N}}$ there must be some i such that $p_i \in W$, in which case we say that $\mathbb{P}$ *meets* W.

For most computable orders of interest, there cannot exist a computable generic sequence. However, there is a way to probabilistically obtain one, using the so-called fireworks technique. This was first proven by Kurtz [8] who showed that one can probabilistically obtain a generic sequence when $\mathbb{P}$ is the set of strings and $\sigma \leq \tau$ when τ is a prefix of σ (Kurtz himself drew upon an argument of Martin [9] who had shown that one can probabilistically construct a hyperimmune set). The probabilistic nature of Kurtz's and Martin's arguments was somewhat hidden in their proof (they used a different framework sometimes referred to as "risking measure"). Rumyantsev and Shen [13] simplified Kurtz's presentation of this technique (although they only focused on Martin's result about hyperimmunity) by giving an explicit probabilistic algorithm. They illustrated their algorithm by a metaphor about a buyer who tries to buy fireworks in a shop, hence the name. Shen and Rumyantsev's presentation allowed Bienvenu and Patey [2, Section 1.4] to make the following generalization to any computable order.

▶ **Theorem 15** (Fireworks master theorem [2]). *For any computable order $\mathbb{P}$, there exists a Turing functional Φ with range $\mathbb{P}$ such that for a set of Z's of positive measure, we have that $\Phi^Z(i)$ is defined for all i and the sequence $(\Phi^Z(i))_{i\in\mathbb{N}}$ is generic.*

For our proof of Theorem 8, we are going to use the order $\mathbb{P}$ whose elements are finite approximations of martingales with positive rational values. Specifically, a member of $\mathbb{P}$ is a total function f whose domain is $\{0,1\}^{\leq n}$ for some n – which we call *length of f* and denote by $lh(f)$ – whose range is $\mathbb{Q}^{>0}$, such that $f(\epsilon) = 1$ and $f(\sigma) = (f(\sigma 0) + f(\sigma 1))/2$ for all σ of length $< lh(f)$. We say that $g \leq f$ if g is an extension of f (i.e., the domain of f is contained in the domain of g and the two coincide on the domain of f). It is clear that $(\mathbb{P}, \leq)$ is a computable order. It is also clear that if $f_1 \geq f_2 \geq \dots$ is a sequence of elements of $\mathbb{P}$ such that $lh(f_i)$ tends to $+\infty$, then $D = \bigcup f_i$ is a total rational valued martingale. This is in particular the case when $(f_i)_{i\in\mathbb{N}}$ is a $\mathbb{P}$-generic sequence, because for every n, the set of elements of $\mathbb{P}$ of length at least n is dense; in this case, we say that the martingale $D = \bigcup f_i$ is a *$\mathbb{P}$-generic martingale.*

▶ **Lemma 16.** *Let D be a $\mathbb{P}$-generic martingale. For every computable sequence A and integer k there exists s such that D reaches capital at least k while playing against the prefix of A of length s (that is, $D(A \upharpoonright l) > k$ for some $l < s$).*

Proof. Fix a computable A and consider the set

$$W = \{g \in \mathbb{P} \mid (\exists l)\ g(A \upharpoonright l) > k\}.$$

We claim that W is a dense c.e. subset of $\mathbb{P}$. That it is c.e. is clear. Now, take any $f \in \mathbb{P}$. Let $n = lh(f)$. By definition of $\mathbb{P}$, $f(A \upharpoonright n)$ is positive, so we can pick an $m > n$ such that $2^{m-n} \cdot f(A \upharpoonright n) > k$. Let g be the martingale of length m which behaves like f up to length n and after that stage plays the doubling strategy on A (and stops betting outside of A). Formally:

$$g(\tau) = \begin{cases} f(\tau) & \text{if } |\tau| \leq n \\ f(\tau \upharpoonright n) & \text{if } |\tau| \geq n \text{ and } \tau \upharpoonright n \neq A \upharpoonright n \\ 0 & \text{if } \tau \upharpoonright n = A \upharpoonright n \text{ but } \tau \text{ is not a prefix of } A \\ f(A \upharpoonright n) \cdot 2^{|\tau|-n} & \text{if } \tau \text{ is a prefix of } A. \end{cases}$$

It is easy to check that g is a finite approximation of martingale which extends f and by construction $g(A \upharpoonright m) = 2^{m-n} \cdot f(A \upharpoonright n) > k$. Thus W is indeed dense. ◀

We can now finish the proof of our main result.

Proof of Theorem 8. By Theorem 15 applied to our partial order $(\mathbb{P}, \leq)$, there is a Turing functional Φ and a set $\mathcal{G}$ of positive measure such that for every $Z \in \mathcal{G}$, $\Phi^Z(n)$ is a $\mathbb{P}$-generic sequence. Thus for $Z \in \mathcal{G}$, $D^Z = \bigcup_n \Phi^Z(n)$ is a $\mathbb{P}$-generic martingale.

Let A be a computable sequence and e be a code for A. By Lemma 16, for every $Z \in \mathcal{G}$, there exists some l_e^Z such that D^Z – being a $\mathbb{P}$-generic martingale – reaches capital at least e at some point while playing against the prefix $A \upharpoonright l_e^Z$.

Now, for each e which is the code of a computable sequence choose some s_e large enough to have

$$\mu\{Z \in \mathcal{G} \mid l_e^Z \leq s_e\} \geq (1 - 2^{-e-1})\mu(\mathcal{G})$$

(and for e which is not a code for a computable sequence, choose s_e arbitrarily).

This guarantees that

$$\mu\{Z \in \mathcal{G} \mid (\forall e \text{ code for a computable seq.})\ l_e^Z \leq s_e\} \geq \mu(\mathcal{G})/2 > 0.$$

Let $\mathcal{H}$ be the set of the left-hand side of this inequality.

Let us consider the sequence $\Delta((s_e)_{e\in\mathbb{N}})$, which by construction is partial computably random. For every $Z \in \mathcal{H}$, for every computable sequence A of code e, the martingale D^Z reaches capital at least e on $A \upharpoonright s_e$. On the other hand, by Fact 1, we know that for infinitely many e, the sequence $\Delta((s_e)_{e\in N})$ coincides with the computable sequence A of index e on a prefix of length $\geq s_e$. Thus this guarantees that for $Z \in \mathcal{H}$, D^Z reaches capital at least e while playing on $\Delta((s_e)_{e\in\mathbb{N}})$. Thus $\Delta((s_e)_{e\in\mathbb{N}})$ is partial computably random but not almost everywhere computably random since $\mathcal{H}$ has positive measure. ◀

4 Conclusion and open questions

In this paper, we have compared the power of deterministic and probabilistic prediction. To this end, we have introduced two notions – a.e. partial computable randomness and a.e. computable randomness. In contrast with Buss and Minnes' results [4], where (due to the stronger limitations on the class of martingales considered) the authors obtained equivalent characterizations of partial computable and computable randomness in terms of probabilistic martingales, our notions do not correspond to their deterministic counterparts, but are, indeed, strictly stronger. The following diagram summarizes the mutual relationships between these notions.

$$\begin{tikzcd}
 & \text{PCR} \arrow[dr, phantom, sloped, "\subsetneq"] & \\
\text{a.e.PCR} \arrow[ur, phantom, sloped, "\subsetneq"] \arrow[dr, phantom, sloped, "\subsetneq"] & \not\subseteq \quad \not\supseteq & \text{CR} \\
 & \text{a.e.CR} \arrow[ur, phantom, sloped, "\subsetneq"] &
\end{tikzcd}$$

The main results of this paper, in fact, concern the incomparability of the notions of a.e. computable randomness and partial computable randomness: on the one hand, by Theorem 8, partial computable randomness does not imply a.e. computable randomness; on the other hand, Theorem 10 states that every a.e. dominating degree computes (actually, contains) a sequence which is a.e. computably random but not partial computably random.

We conclude this paper by pointing out interesting further directions to be investigated on this topic.

The main goal we have achieved is the construction of a partial computable random sequence X which is not a.e. computably random: from the perspective of algorithmic randomness, this amounts to say that any sufficiently random sequence Z derandomizes X, in the sense that X is not computably random relative to Z. But how much randomness is actually needed to derandomize such a sequence? In particular, is Martin-Löf randomness enough? In this regard, we ask the following question.

▶ **Question 1.** *Given a partial computably random sequence X which is not a.e. computably random, can there be a Martin-Löf random sequence Z such that X is still computably random relative to Z? If so, is there always such a Z?*

The second open question is more general, and strongly related with one of the main theoretical motivations leading to this work, namely the failure of the analogue of van Lambalgen's theorem for computable randomness. Theorem 8, in fact, can be regarded as a strong failure of this result for computable randomness, because of the existence of computably random sequences that, nevertheless, can be derandomized by almost every oracle. It is known that the analogue of van Lambalgen's theorem fails for other randomness notions studied in the literature, such as Schnorr randomness, Kurtz randomness and Demuth randomness (see [7]). However, we do not know if it fails in the strong sense mentioned above.

▶ **Question 2.** *Are there other randomness notions for which an analogue of Theorem 8 holds (namely, for which there is a random sequence which is not a.e. random)?*

In particular, it seems that our constructions may be easily modified to get results about a.e. Schnorr randomness.

References

1 Laurent Bienvenu. *Game-theoretic characterizations of randomness: unpredictability and stochasticity.* PhD thesis, Université de Provence, 2008.

2 Laurent Bienvenu and Ludovic Patey. Diagonally non-computable functions and fireworks. *Information and Computation*, 253:64–77, 2017.

3 Stephen Binns, Bjørn Kjos-Hanssen, Manuel Lerman, and Reed Solomon. On a conjecture of Dobrinen and Simpson concerning almost everywhere domination. *Journal of Symbolic Logic*, 71(1):119–136, 2006.

4 Sam Buss and Mia Minnes. Probabilistic algorithmic randomness. *Journal of Symbolic Logic*, 78(2):579–601, 2013.

5 Karel de Leeuw, Edward F. Moore, Claude Shannon, and Norman Shapiro. Computability by probabilistic machines. In *Automata Studies*. Princeton University Press, 1956.

6 Natasha Dobrinen and Stephen G. Simpson. Almost everywhere domination. *Journal of Symbolic Logic*, 69(3):914–922, 2004.

7 Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Theory and Applications of Computability. Springer New York, New York, NY, 2010.

8 Stuart Alan Kurtz. *Randomness and Genericity in the Degrees of Unsolvability.* PhD thesis, University of Illinois at Urbana–Champaign, 1982.

9 Donald Martin. Measure, category, and degrees of unsolvability. Unpublished manuscript, 1967.

10 Wolfgang Merkle. The complexity of stochastic sequences. *Journal of Computer and System Sciences*, 74(3):350–357, 2008.

11 André Nies. *Computability and randomness.* Oxford Logic Guides. Oxford University Press, 2009.

12 André Nies, Frank Stephan, and Sebastiaan Terwijn. Randomness, relativization and Turing degrees. *Journal of Symbolic Logic*, 70:515–535, 2005.

13 Andrei Rumyantsev and Alexander Shen. Probabilistic constructions of computable objects and a computable version of Lovász local lemma. *Fundamenta Informaticae*, 132(1):1–14, 2014.

14 Claus Schnorr. *Zufälligkeit und Wahrscheinlichkeit*, volume 218 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin-Heidelberg-New York, 1971.

15 Michiel van Lambalgen. *Random sequences.* PhD dissertation, University of Amsterdam, Amsterdam, 1987.

16 Liang Yu. When van Lambalgen's theorem fails. *Proceedings of the American Mathematical Society*, 135(3):861–864, 2007.

Single-Source Shortest p-Disjoint Paths: Fast Computation and Sparse Preservers

Davide Bilò
Department of Humanities and Social Sciences, University of Sassari, Italy

Gianlorenzo D'Angelo
Gran Sasso Science Institute, L'Aquila, Italy

Luciano Gualà
Department of Enterprise Engineering, University of Rome "Tor Vergata", Italy

Stefano Leucci
Department of Information Engineering, Computer Science and Mathematics,
University of L'Aquila, Italy

Guido Proietti
Department of Information Engineering, Computer Science and Mathematics,
University of L'Aquila, Italy
Institute for System Analysis and Computer Science "Antonio Ruberti" (IASI CNR), Rome, Italy

Mirko Rossi
Gran Sasso Science Institute, L'Aquila, Italy

Abstract

Let G be a directed graph with n vertices, m edges, and non-negative edge costs. Given G, a fixed source vertex s, and a positive integer p, we consider the problem of computing, for each vertex $t \neq s$, p edge-disjoint paths of minimum total cost from s to t in G. Suurballe and Tarjan [Networks, 1984] solved the above problem for $p = 2$ by designing a $O(m + n \log n)$ time algorithm which also computes a sparse *single-source* 2*-multipath preserver*, i.e., a subgraph containing 2 edge-disjoint paths of minimum total cost from s to every other vertex of G. The case $p \geq 3$ was left as an open problem.

We study the general problem ($p \geq 2$) and prove that any graph admits a sparse single-source p-multipath preserver with $p(n-1)$ edges. This size is optimal since the in-degree of each non-root vertex v must be at least p. Moreover, we design an algorithm that requires $O(pn^2(p + \log n))$ time to compute both p edge-disjoint paths of minimum total cost from the source to all other vertices and an optimal-size single-source p-multipath preserver. The running time of our algorithm outperforms that of a natural approach that solves $n-1$ single-pair instances using the well-known *successive shortest paths* algorithm by a factor of $\Theta(\frac{m}{np})$ and is asymptotically near optimal if $p = O(1)$ and $m = \Theta(n^2)$. Our results extend naturally to the case of p vertex-disjoint paths.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases multipath spanners, graph sparsification, edge-disjoint paths, min-cost flow

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.12

Funding This work has been partially supported by the Italian MIUR PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Markets".

1 Introduction

Consider a communication network modelled as a directed graph G with n vertices, m edges, and non-negative edge costs. Whenever a *source* vertex s needs to send a message to a *target* vertex t, we are faced with the problem of finding a *good* path connecting s and t in G. Typically, this path is chosen with the aim of minimizing the communication cost, i.e., the

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 12; pp. 12:1–12:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

sum of the costs of the path's edges. In a scenario where some edges of the network might be congested or faulty, it is useful to introduce some degree of redundancy in order to improve the communication reliability. One of the possible approaches that aims to formalize the above requirements asks to find p edge-disjoint paths from s to t in G for some integer $p \geq 2$. Quite naturally, similarly to the case of the shortest path, we would like to minimize the sum of the costs of the edges in the selected paths.

This is equivalent to the problem of computing a minimum-cost flow of value p from s to t in the unit-capacity network G and can be solved in time $O(p(m + n \log n))$ using the *successive shortest paths* (SSP) algorithm [2, 14].

In this paper we focus on the *single-source* case, in which a fixed source vertex s wants to communicate with every other vertex t using p edge-disjoint paths. We distill the above discussion into the following two problems:

Single-source p-multipath preserver problem: We want to find a *sparse* subgraph H of G such that, for every vertex $t \neq s$, H contains p edge-disjoint paths of minimum total cost from s to t in G. We will refer to such a subgraph H as a *single-source p-multipath preserver*. Among all possible feasible solutions, we aim at computing the one of minimum *size*, i.e., having the minimum number of edges.

Shortest p edge-disjoint paths problem: For every vertex $t \neq s$, we want to compute a subset S^t of edges from G that induce p edge-disjoint paths of minimum total cost from s to t in G.

Observe that if the graph G is not sufficiently connected, the single-source p-multipath preserver H and some of the sets S^t defined above might not exist. To avoid this issue, we assume that G is p-edge-outconnected from s, i.e., given any vertex $t \neq s$, G contains p edge-disjoint paths from s to t.[1]

The above problems have been addressed by Suurballe and Tarjan for the special case $p = 2$ in [25], where they provide an algorithm requiring time $O(m + n \log n)$ to compute both a single-source p-multipath preserver of size $2(n - 1)$ and (a compact representation of) all sets S^t of the shortest p edge-disjoint paths problem.[2] In their paper, the authors mention the case $p > 2$ as an important open problem.

In this paper we provide the following results:

- We prove in Section 3 that any graph G always admits a single-source p-multipath preserver of size $p(n - 1)$. This size is optimal since the in-degree of each non-source vertex t in H needs to be at least p, even to preserve the p-edge-outconnectivity from s to t.
- In Section 4 we design an algorithm that requires $O(pn^2(p + \log n))$ time to solve the shortest p edge-disjoint paths problem. This improves over the natural algorithm that computes the sets S^t with $n-1$ independent invocations of the SSP algorithm, which would require $O(pnm + pn^2 \log n)$ time. Up to logarithm factors, our algorithm is $\Theta(\frac{m}{np})$ times faster than the above algorithm based on SSP. Moreover, for $p = O(1)$ and $m = \Theta(n^2)$, the time complexity of our algorithm is optimal up to logarithmic factors. Finally, our algorithm also computes a single-source p-multipath preserver H of optimal size that contains all the edges in the sets S^t.

We point out that a modification of our algorithm allows us to handle graphs G that are not p-edge-outconnected from s. In this case, our algorithm computes, for each vertex $t \neq s$, a set of edges S^t that induce $\sigma(t)$ edge-disjoint paths from s to t of minimum total cost in G,

[1] It is possible to check whether a graph is p-edge-outconnected from s in $O(pm \log \frac{n^2}{m})$ time [16].
[2] After the execution of their algorithm, it is possible to compute each set S^t in time $O(|S^t|)$.

where $\sigma(t)$ is the minimum between p and the maximum number of edge-disjoint paths from s to t. Moreover, the algorithm also returns a subgraph H of G of optimal size where each vertex $t \neq s$ has exactly $\sigma(t)$ incoming edges. As in the previous case, H contains all edges in the sets S^t. The running time of our algorithm is asymptotically unaffected.

We also discuss a variant of the problem in which, instead of minimizing the overall cost of p edge-disjoint paths, we aim at minimizing the cost of the path with maximum cost. We show that our algorithm provides an optimal p-approximation, unless $\mathrm{P} = \mathrm{NP}$.

Finally, our results can be extended to the case of vertex-disjoint paths and to undirected graphs via standard transformations of the input graph, thus proving an upper bound of $p(n-1)$ on the size of p-multipath preservers in undirected graphs, which was posed as an open problem in [17]. All the above modifications and variants are discussed in Appendix A.

Related work. As already mentioned, the closest related work is the paper by Suurballe and Tarjan that studies the case $p = 2$ [25].

The single-source p-multipath preserver problem falls within a broad class of problems with a long-standing research tradition. Here we are given a graph G and we want to select a sparse subgraph H of G which maintains, either in an exact or in an approximate sense, some distance-related property of interest. The goal is that of understanding the best trade-offs that can be attained between the size of H and the accuracy of the maintained properties. As a concrete example, if we focus on the cost of a single path (i.e., $p = 1$) between pairs of vertices, a well-known notion adopted is that of *graph spanners*, which has been introduced by Peleg and Schäffer [24]. A spanning subgraph H is an α-spanner of G if the distance of each pair of vertices in H is at most α times the corresponding distance in G. If G is undirected then it is possible to compute, for any integer $k \geq 1$, a $(2k-1)$-spanner of size $O(n^{1+\frac{1}{k}})$ [3] (if we assume the Erdős Girth Conjecture [13], this trade-off is asymptotically optimal), while there exist directed graphs for which any α-spanner has size $\Omega(n^2)$.

When $\alpha = 1$ and hence H retains the exact distances of G, a 1-spanner is usually called a *preserver*. While $\Omega(n^2)$ edges might be necessary to preserve all-to-all distances, better trade-offs can be obtained if we only care to preserve distances between some pairs of vertices. For example, a shortest-path tree can be seen as a sparse *single-source* preserver. More significant trade-offs can be obtained for different choices of the pairs of interest (see, e.g., [8, 10]). For more related results on the vast area of spanners and preservers, we refer the interested reader to the survey in [1].

Concerning the case of multiple paths ($p > 1$), Gavoille et al. [17] introduced the notion of *p-multipath spanner* of a weighted graph G, from which we borrow the term *multipath*. A p-multipath α-spanner of G is a spanning subgraph H of G containing, for each pair of vertices u, v, p edge-disjoint paths from u to v of total cost at most α times the cost of the cheapest p edge-disjoint paths from u to v in G. Among other results, the authors of [17] prove the existence, for any choice of p and for any $k \geq 1$, of a p-multipath $p(2k-1)$-spanner of size $O(pn^{1+\frac{1}{k}})$ for undirected graphs. Following [17], there has been further work on multipath spanners [11, 18]. All of the above papers, however, focus on approximated costs, in the all-pairs setting on undirected graphs. Since our focus is on p-multipath α-spanners for directed graphs, in the single-source case, and for $\alpha = 1$, such results cannot be directly compared to the one in this work.

As discussed above, edge-disjoint paths can be seen as a strategy to achieve *fault-tolerance* through redundancy. In particular, Baswana et al. introduced the problem of computing a sparse *k-fault tolerant reachability subgraph* H of a given directed graph G, i.e., a subgraph that preserves reachability from a distinguished source vertex s following the failure of *any*

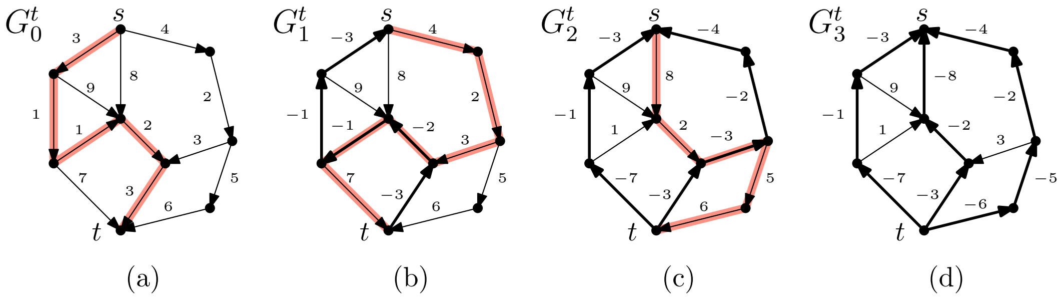


Figure 1 An execution of the successive shortest paths algorithm on a graph G. Figures (a), (b), (c), and (d) respectively show the residual networks $G_0^t = G$, G_1^t, G_2^t, and G_3^t. The shortest paths computed by the algorithm are highlighted in red. The edges that appear in the opposite orientation w.r.t. G are shown in bold. If we orient the bold edges in G_i^t as in G, we obtain the edges in S_i^t, i.e., those belonging to i edge-disjoint paths of minimum total cost from s to t in G.

set of at most k edges [5]. The authors show that it is always possible to select a subgraph H with $O(2^k n)$ edges, and that this bound is tight in the sense that there are graphs G with $\Omega(2^k n)$ edges that cannot be sparsified.

We remark that the results for the problem in [5] are not directly comparable to ours. Indeed, when G is not p-edge-outconnected from s, a subgraph containing $\sigma(t)$ edge-disjoint paths from s to every other vertex t is not necessarily a $(p-1)$-fault tolerant reachability subgraph. As a consequence, the lower bound of $\Omega(2^k n)$ does not apply to our problem. On the other hand, when we restrict ourselves to graphs that are p-edge-outconnected from s, any single-source p-multipath preserver is also a $(p-1)$-fault tolerant reachability subgraph, yet the converse is not true. Indeed, a $(p-1)$-fault tolerant reachability subgraph does not necessarily guarantee that the cost of the p edge-disjoint paths from s to each t is minimized.

Other approaches to address faults in networks, which aim at (approximately) preserving the length of the surviving shortest paths from a source vertex s, are captured by the notion of single-source fault-tolerant spanners and preservers [4, 6, 7, 9, 19–23].

2 Preliminaries

We denote by $V(G)$, $E(G)$, and $c : E(G) \to \mathbb{R}^+$, the set of vertices, the set of edges, and the cost function, respectively. With a slight abuse of notation, if S is a set of edges (resp. π is a path), we denote by $c(S)$ (resp. $c(\pi)$) the sum of the costs $c(e)$ for $e \in S$ (resp. $e \in E(\pi)$).

In order to lighten the notation, in the rest of the paper we will assume that the graph is anti-symmetric, i.e., if $(u, v) \in E(G)$, then $(v, u) \notin E(G)$. We make this assumption as we will define auxiliary graphs on the vertex set $V(G)$ in which some edge $(u, v) \in E(G)$ might appear in the reversed direction (v, u) and therefore, a non anti-symmetric graph G may cause the presence of two parallel edges in the auxiliary graphs. It is easy to remove this assumption by distinguishing the two possible parallel edges with unique identifiers.

Relation with the s-t-min-cost flow problem. For a fixed pair of vertices $s, t \in V(G)$, the problem of finding p edge-disjoint paths of minimum total cost from s to t is a special case of the *s-t-min-cost flow problem* where edges have unit capacities and the goal is to send p units of flow from s to t at minimum total cost. *Successive shortest path* (SSP) [2, 14] is a well-known algorithm that solves the s-t-min-cost flow problem. We now give a brief description of SSP for the special case of unit edge capacities.

The algorithm sends p units of flow from s to t by iteratively pushing one new unit of flow through a shortest path from s to t in the *residual network* associated with the current flow. More precisely, let the initial residual network be $G^t_0 = G$. In the generic i-th iteration, SSP finds a shortest path π_t from s to t in G^t_{i-1}, and uses it to compute a residual network G^t_i.

The residual network G^t_i is obtained from G^t_{i-1} by *reversing* all the edges in π_t, where *reversing* an edge (u, v) of cost $c(u, v)$ means replacing (u, v) with the edge (v, u) of cost $c(v, u) = -c(u, v)$. See Figure 1 for an example.

At the end of the i-th iteration, the i units of flow are sent through the edges of G that are reversed in the residual network G^t_i. We denote by S^t_i the set of such edges, which contains exactly the edges of i edge-disjoint paths from s to t of minimum total cost. Therefore, once the p-th iteration is completed, S^t_p is a solution for the problem of finding p edge-disjoint paths of minimum total cost from s to t. An interesting observation that we will use later on is the following.

▶ Remark 1. The set S^t_i can be computed from S^t_{i-1} and π_t in $O(|S^t_i| + n)$ time by first setting $S^t_i = S^t_{i-1}$ and then by (i) deleting from S^t_i all edges $(u, v) \in S^t_i$ that are reversed in $E(\pi_t)$, and (ii) adding to S^t_i all edges $(u, v) \in E(\pi_t) \cap E(G)$.

A straightforward implementation of the above algorithm requires time $O(pnm)$ since it computes p shortest paths using the Bellman-Ford algorithm (notice, indeed, that the edge costs in the residual networks might be negative). The above time complexity can be improved to $O(p(m + n \log n))$ by suitably re-weighting the residual network so that edge costs are non-negative and shortest paths are preserved, allowing the Dijkstra algorithm to be used in place of Bellman-Ford [14].

We can solve $n - 1$ separated instances of s-t-min-cost flow (one for each node t) and obtain (i) the solution for the shortest p edge-disjoint paths problem, i.e., the sets S^t_p for each $t \in V(G) \setminus \{s\}$; (ii) a single-source p-multipath preserver by making the union of all solutions S^t_p obtained. However, the resulting single-source p-multipath preserver may not be sparse and the total running time needed to solve both problems is $O(pn(m + n \log n))$.

In Section 3 we show the existence of a single-source p-multipath preserver of optimal size $p(n - 1)$ and in Section 4 we design an algorithm that solves both our problems in time $O(pn^2(p + \log n))$.

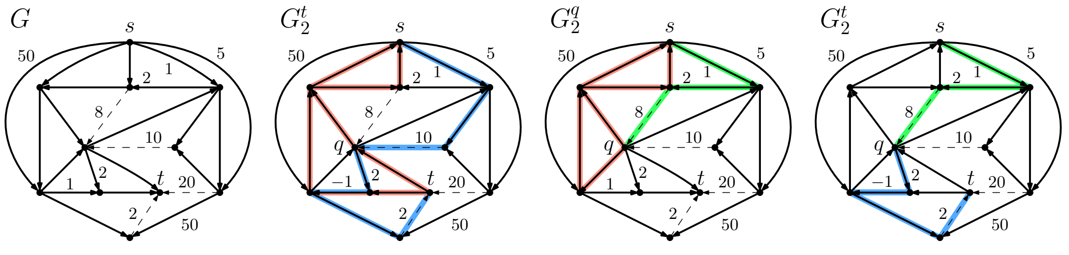
3 An optimal-size single-source p-multipath preserver

In this section we show that it is possible to compute a single-source p-multipath preserver having size $p(n - 1)$.

We compute such a preserver iteratively: we start with an empty graph $H_0 = (V(G), \emptyset)$ and, during the i-th iteration, we construct a i-multipath preserver[3] H_i of G by adding to H_{i-1} a single new edge e_t entering in t for each vertex $t \in V(G) \setminus \{s\}$.[4] This process stops at the end of the p-th iteration. We will show that H_p is a sparse single-source p-multipath preserver. Notice that, by construction, vertex s has in-degree 0 in H_i and each other vertex has in-degree i, therefore H_p has size $p(n - 1)$.

[3] In the following we might shorten *single-source i-multipath preserver* to *i-multipath preserver* or, when i is clear from the context, simply *preserver*.

[4] For a fixed vertex $t \neq s$, the edge e_t will be the last edge of a path from s to t in a suitable graph. Hence, in each iteration, we augment the preserver with the union of all such last edges, in a way that resembles the techniques used in [23, 24].



(a) (b) (c) (d)

Figure 2 An example of the suboptimality property of Lemma 5 for $i = 3$. (a) The graph G, in which the edges in H_2 are solid and the edges in $E(G) \setminus E(H_2)$ are dashed. Unlabelled edges cost 0. For graphical convenience, G is 2-edge-outconnected from s and can be made 3-edge-outconnected by a suitable addition of costly edges. (b) The graph G_2^t in which the edges that appear in S_2^t in their opposite orientation are highlighted in red. A shortest path $\pi_t \in \Pi(s, t; G_2^t)$ is highlighted in blue and traverses vertex $q = q(\pi_t)$. (c) The graph G_2^q where the (reversed versions of) the edges in S_2^q are highlighted in red, and a shortest path $\pi_q \in \Pi(s, q; G_2^q)$ is highlighted in green. (d) The graph G_2^t where the path $\pi_q \circ \pi_t[q : t]$ is highlighted in green and blue and belongs to $\Pi(s, t; G_2^t)$.

We will prove by induction on i that H_i contains i edge-disjoint paths of minimum total cost (in G) from s to all vertices $t \in V(G) \setminus \{s\}$. Since this is trivially true when $i = 0$, in the rest of the section we assume that the induction hypothesis is true for H_{i-1} with $1 \leq i < p$, and we focus on proving that it remains true for H_i.

Following the notation of Section 2, we denote by S_{i-1}^t the set of edges belonging to any $i-1$ edge disjoint paths from s to t of minimum total cost in H_{i-1} (and hence in G). We let G_{i-1}^t be the residual network obtained from G by reversing the edges in S_{i-1}^t.

To prove that H_p is a single-source p-multipath preserver we need to employ a suitable tie-breaking rule between paths of the same cost. Although randomly perturbing the edge-weights would be sufficient to prove the main result of this section, we will instead introduce a different tie-breaking scheme. We use this scheme to provide structural lemmas that will also be used in Section 4 to prove the correctness of our time-efficient deterministic algorithm that computes a p-multipath preserver and solves the shortest p edge-disjoint paths problem.

To this aim, we define distances as pairs of elements from $\mathbb{R} \cup \{+\infty\}$. Given $d = (d_1, d_2)$ and $d' = (d'_1, d'_2)$ we denote by $d + d'$ the pair $(d_1 + d'_1, d_2 + d'_2)$. We also compare distances lexicographically, and write $d \prec d'$ to denote that the pair d precedes d' in the lexicographical order. Similarly, $d \preceq d'$ if $d \prec d'$ or $d = d'$. Given any path π, let $\eta(\pi)$ be the number of edges of π that are in $E(G) \setminus E(H_{i-1})$. We can associate π with a pair $|\pi| = (c(\pi), \eta(\pi))$. With a slight abuse of notation, we can therefore extend the above linear order to paths: for two paths π and π', we write $\pi \prec \pi'$ (resp. $\pi \preceq \pi'$) as a shorthand for $|\pi| \prec |\pi'|$ (resp. $|\pi| \preceq |\pi'|$). Intuitively, when we compare paths w.r.t. $\preceq$, the values of $\eta(\cdot)$ serve as tie-breakers between paths having the same cost. In the following $\Pi(u, v; G')$ will denote the set of paths from u to v in G' that are shortest w.r.t. the total order relation $\preceq$. When $\Pi(u, v; G')$ contains a single path we denote by $\pi(u, v; G')$ the sole path in $\Pi(u, v; G')$. Given a path π_1 from v_0 to v_1 and a path π_2 from v_1 to v_2, we denote by $\pi_1 \circ \pi_2$ the path from v_0 to v_2 that is obtained by composing π_1 and π_2. Given a path π from v_0 and v_1, and two distinct vertices u and v of π such that π traverses u and v in this order, we denote by $\pi[u : v]$ the subpath of π from u to v.

The edge e_t entering t selected by the algorithm is the last edge of an arbitrarily chosen path in $\Pi(s, t; G_{i-1}^t)$. For $\pi \in \Pi(s, t; G_{i-1}^t)$, we define $q(\pi)$ as the last *internal* vertex of π such that its incoming edge in π belongs to $E(G) \setminus E(H_{i-1})$. If no such vertex exists, we let $q(\pi) = s$ (see Figure 2 (b)).

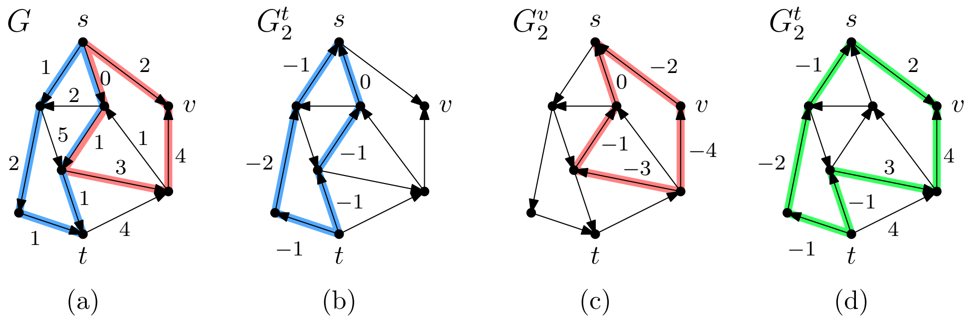


Figure 3 (a) A graph G with non-negative costs. The edges in S_2^t (resp. S_2^v) are highlighted in blue (resp. red) and induce two edge-disjoint paths of minimum total cost from s to t (resp. v) in G. The edges in S_2^v are exactly the ones used by the flow f' in the proof of Lemma 2. (b) The graph G_2^t obtained from G by reversing the edges in S_2^t. The reversed edges (highlighted in blue) correspond to those used by the flow f in the proof of Lemma 2. (c) The graph G_2^v obtained from G by reversing the edges in S_2^v. The reversed edges are highlighted in red. (d) The graph G_2^t in which the edges in $\Delta(t,v)$ are highlighted in green and induce 2 edge-disjoint paths of minimum total cost from t to v in G_2^t. These edges are the ones used by the flow f'' in the proof of Lemma 2. The edge costs that are missing in (b), (c), or (d) match those of the corresponding edges in (a).

The main technical ingredient of the result in this section is a suboptimality property, which will be given formally in Lemma 5. Intuitively, if $q = q(\pi_t)$ for some path $\pi_t \in \Pi(s,t;G_{i-1}^t)$, then this property ensures that the composition $\pi_q \circ \pi_t[q:t]$ of any shortest path $\pi_q \in \Pi(s,q;G_{i-1}^q)$ with the suffix $\pi_t[q:t]$ of π_t is also a shortest path in $\Pi(s,t,G_{i-1}^t)$. Since (up to the orientation of its edges) $\pi_t[q:t]$ contains a single edge e_t not already in H_{i-1} (i.e., the one entering in t), this property allows to reuse the edges in H_{i-1} and in $S_i^q \setminus E(H_{i-1})$ to build $S_i^t \setminus \{e_t\}$. See Figure 2 (d) for an example. The rest of this section formalizes the above intuition.

Given any two nodes $t, v \in V(G)$, we denote by $\Delta(t,v)$ the set of edges of G_{i-1}^t that appear in the opposite orientation in G_{i-1}^v (see Figure 3.). Formally, $(x,y) \in \Delta(t,v)$ iff $(x,y) \in E(G_{i-1}^t)$ and $(y,x) \in E(G_{i-1}^v)$. As a consequence, $(x,y) \in \Delta(t,v)$ iff $(y,x) \in \Delta(v,t)$. Moreover, we observe that $(x,y) \in \Delta(t,v)$ iff exactly one of the following conditions holds: (i) $(y,x) \in S_{i-1}^t$; (ii) $(x,y) \in S_{i-1}^v$. Finally, we notice that $E(G) \cap \Delta(t,v) \subseteq S_{i-1}^v \subseteq E(H_{i-1})$. The next three lemmas will be instrumental to prove Lemma 5.

▶ **Lemma 2.** *The edges in $\Delta(t,v)$ are exactly those belonging to $i-1$ edge disjoint paths of minimum total cost from t to v in G_{i-1}^t.*

Proof. Consider G_{i-1}^t as an instance of min-cost flow with unit capacity where we want to send $i-1$ units of flow from t to v. We define a first flow assignment f that sends $i-1$ units of flow from t to s in G_{i-1}^t using the edges in S_{i-1}^t in the reverse direction (see Figure 3 (b)). More precisely, $\forall (x,y) \in E(G_{i-1}^t)$, $f(x,y) = 1$ if $(y,x) \in S_{i-1}^t$, and $f(x,y) = 0$ otherwise. Notice that f is a flow of value $|f| = i-1$ in G_{i-1}^t and that the associated residual graph is G. We now consider a minimum-cost flow f' that pushes $i-1$ units of flow from s to v in G using the edges in S_{i-1}^v (see Figure 3 (a) where the edges used by f' are highlighted in red). In particular, we define $f'(e) = 1$ if $e \in S_{i-1}^v$, and $f(e) = 0$ otherwise. The residual graph associated with f' (w.r.t. G) is G_{i-1}^v and, since f' is a minimum-cost flow, G_{i-1}^v does not contain any negative-cost cycle [14].

We can obtain a flow f'' from t to v in G_{i-1}^t with $|f''| = i-1$ by composing f and f': we first push $i-1$ units of flow from t to s in G_{i-1}^t according to f and then push $i-1$ units of flow from s to t in the residual network G according to f' (see Figure 3 (d)). More precisely,

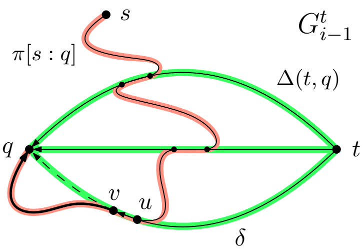


Figure 4 A qualitative representation of the proof of Lemma 3. We are supposing towards a contradiction that $\pi[s:q]$ (highlighted in red) is not entirely contained in G^q_{i-1} and hence traverses an edge (u,v) belonging to the set $\Delta(t,q)$ (highlighted in green). The subpath of π (resp. δ) from v to q is shown in bold (resp. is dashed).

the resulting net flow f'' is defined as follows: given $(x,y) \in E(G^t_{i-1})$, $f''(x,y) = 1$ iff either (i) $f(x,y) = 1$ and $f'(y,x) = 0$, or (ii) $f(x,y) = 0$ and $f'(x,y) = 1$. The residual network associated with f'' (w.r.t. G^t_{i-1}) is exactly G^v_{i-1} and, since it contains no negative-cost cycles, f'' is also a minimum-cost flow.

To conclude the proof it suffices to notice that the edges (x,y) for which $f''(x,y) = 1$ are exactly those in $\Delta(t,v)$. ◀

▶ **Lemma 3.** *For every $t \in V(G)$, let $\pi \in \Pi(s,t;G^t_{i-1})$ and $q = q(\pi)$. The subpath $\pi[s:q]$ is entirely contained in G^q_{i-1}.*

Proof. If $s = q$ the subpath $\pi[s:q]$ is empty and the claim is trivially true. We therefore consider $s \neq q$ and suppose towards a contradiction that $\pi[s:q]$ is not entirely contained in G^q_{i-1}. Then, $\pi[s:q]$ traverses at least one edge in $\Delta(t,q)$. Let (u,v) be the last edge traversed by $\pi[s:q]$ that belongs to $\Delta(t,q)$. By Lemma 2, the edges in $\Delta(t,q)$ induce $i-1$ edge disjoints paths of minimum total cost from t to q in the subgraph of G^t_{i-1}. Let δ one such such path traversing (u,v).

Since, by definition of q, the edge e of $\pi[s:q]$ entering in q is in $E(G) \setminus E(H_{i-1})$, we have $e \notin \Delta(t,q)$. Then, the subpath $\pi[v:q]$ of $\pi[s:q]$ is not empty and, by our choice of (u,v) does not traverse any edge in $\Delta(t,q)$.

By the suboptimality property of shortest paths, $\pi[v:q] \preceq \delta[v:q]$ and hence $c(\pi[v:q]) \leq c(\delta[v:q])$. If $c(\pi[v:q]) < c(\delta[v:q])$, we can replace $\delta[v:q]$ with $\pi[v:q]$ in δ to obtain a path δ' from t to q in G^t_{i-1} with $c(\delta') < c(\delta)$. This contradicts Lemma 2 since it implies the existence of $i-1$ edge-disjoint paths from t to q in G^t_{i-1} with a total cost smaller than $c(\Delta(t,q))$ (see Figure 4).

If $c(\pi[v:q]) = c(\delta[v:q])$, we can replace $\pi[v:q]$ with $\delta[v:q]$ in $\pi[s:q]$ to obtain a path π' from s to q in G^t_{i-1} satisfying $c(\pi') = c(\pi[s:q])$. Since all edges of $E(G) \cap E(\delta[v:q]) \subseteq \Delta(t,v)$ are in H_{i-1}, $\pi[v:q]$ contains more edges in $E(G) \setminus E(H_{i-1})$ than $\delta[v:q]$, thus $\pi' \prec \pi[s:q]$. This is a contradiction since, by the suboptimality property of shortest paths and by our choice of $\pi \in \Pi(s,t;G^t_{i-1})$, $\pi[s:q]$ must be a shortest path from s to q in G^t_{i-1} w.r.t. $\preceq$. ◀

▶ **Lemma 4.** *Let $t,q \in V(G) \setminus \{s\}$, and let π be a simple path from s to q in G^q_{i-1} such that the edge of π entering in q is in $E(G) \setminus E(H_{i-1})$. If π is not entirely contained in G^t_{i-1}, then there exists a path π' from s to q in G^t_{i-1} such that $\pi' \prec \pi$.*

Proof. If π is not entirely contained in G^t_{i-1} then π traverses some edge in $\Delta(q,t)$. Consider the first edge $(u,v) \in \Delta(q,t)$ traversed by π, and let δ be a simple path, from q to t that traverses (u,v) in the subgraph of G^q_{i-1} induced by $\Delta(q,t)$. Since in G^q_{i-1} there are no negative

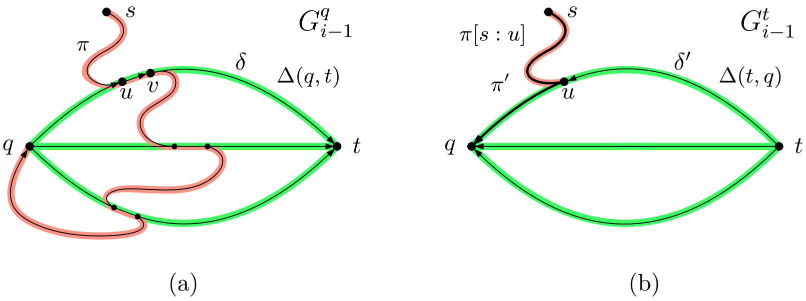


Figure 5 A qualitative representation of the proof of Lemma 4. We are assuming that π (highlighted in red) is a path from s to q in G^q_{i-1} entering in q with and edge of $E(G) \setminus E(H_{i-1})$. (a) π intersects a path δ among the $i-1$ edge-disjoint paths induced by the edges in $\Delta(q,t)$ (highlighted in green). (b) The edges of path δ' obtained by reversing the edges in δ belong to $\Delta(t,q)$ (highlighted in green). Then, the path $\pi' = \pi[s:u] \circ \delta'[u:q]$ (shown in bold) is entirely contained in G^t_{i-1} and satisfies $\pi' \prec \pi$.

cycles [14], we have that $c(\pi[u:q]) + c(\delta[q:u]) \geq 0$ and hence $c(\pi[u:q]) \geq -c(\delta[q:u])$. By reversing the edges in the subpath $\delta[q:u]$ we obtain a path δ' from u to q that uses only edges in $\Delta(t,q)$ and has cost $c(\delta') = -c(\delta(u,q))$. We can then select $\pi' = \pi[s:u] \circ \delta'$. Notice indeed that $c(\pi') = c(\pi[s:u]) + c(\delta') = c(\pi[s:u]) - c(\delta[q:u]) \leq c(\pi[s:u]) + c(\pi[u:q]) = c(\pi)$ (see Figure 5). Moreover, δ' does not use any edge in $E(G) \setminus E(H_{i-1})$ while the last edge in $\pi[u:q]$ is in $E(G) \setminus E(H_{i-1})$. This shows that $\pi' \prec \pi$ and concludes the proof. ◀

▶ **Lemma 5** (Suboptimality property). *Fix $t \in V(G)$, let $\pi_t \in \Pi(s,t;G^t_{i-1})$, $q = q(\pi_t)$, and $\pi_q \in \Pi(s,q;G^q_{i-1})$. We have that $\pi_q \circ \pi_t[q:t] \in \Pi(s,t;G^t_{i-1})$.*

Proof. We start by showing that π_q must be entirely contained in G^t_{i-1}. To this aim suppose towards a contradiction that π_q is not entirely contained in G^t_{i-1}. By Lemma 4, there exists a path π' in G^t_{i-1} such that $\pi' \prec \pi_q$ and, by Lemma 3, we know that $\pi_t[s:q]$ is entirely contained in G^q_{i-1}. Then, since $\pi_q \in \Pi(s,q,G^q_{i-1})$, we must have $c(\pi_q) \preceq c(\pi_t[s:q])$. We can therefore replace $\pi_t[s,q]$ with π' in π_t and obtain a new path $\pi'' \prec \pi_t$ from s to t in G^t_{i-1}, contradicting $\pi_t \in \Pi(s,t,G^t_{i-1})$.

The path $\pi = \pi_q \circ \pi_t[q:t]$ obtained by replacing $\pi_t[s:q]$ with π_q in π_t is entirely contained in G^t_{i-1} and satisfies $\pi \preceq \pi_t$. Since π_t is a shortest path in G^t_{i-1} w.r.t. $\preceq$, so is π. ◀

Next lemma uses the suboptimality property to show that, for each $t \neq s$, there exists a shortest path δ from s to t in G^t_{i-1} such that, when we orient the edges of δ in the same direction as in G, the resulting set of edges is entirely contained in H_i.

▶ **Lemma 6.** *For each $t \in V(G) \setminus \{s\}$, there exists a path $\delta \in \Pi(s,t,G^t_{i-1})$ such that $E(\delta) \cap E(G) \subseteq E(H_i)$.*

Proof. Define $q_0 = t$. For $j \geq 0$ and $q_j \neq s$, let $\pi_j \in \Pi(s,q_j;G^{q_j}_{i-1})$ be the shortest from s to q_j selected by the algorithm and define $q_{j+1} = q(\pi_j)$ (see Figure 6).

We now show that all q_j are distinct, hence there exists a k for which $q_k = s$. By contradiction, consider the smallest index $j' > j$ such that $q(\pi_{j'}) = q_j$. We will construct two paths towards $q_{j'}$ in $G^{q_{j'}}_{i-1}$ that have different lengths, yet they must both be shortest paths, thus providing the sought contradiction.

By Lemma 5, we know that $\pi_{j+1} \circ \pi_j[q_{j+1}:q_j] \in \Pi(s,q_j;G^{q_j}_{i-1})$. We can repetitively apply Lemma 5, until we get $\delta_j = \pi_{j'} \circ \pi_{j'-1}[q_{j'}:q_{j'-1}] \circ \ldots \circ \pi_j[q_{j+1}:q_j] \in \Pi(s,q_j;G^{q_j}_{i-1})$. Since $q(\pi_{j'}) = q_j$, by Lemma 5 we have that $\delta_{j'} = \delta_j \circ \pi_{j'}[q_j:q_{j'}] \in \Pi(s,q_{j'};G^{q_{j'}}_{i-1})$.

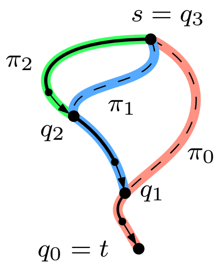


Figure 6 A qualitative representation of the path δ constructed in the proof of Lemma 6 when $k = 3$. The path δ is drawn with solid lines and the portions belonging to $E(H_{i-1})$ are shown in bold. The shortest paths π_0, π_1, and π_2 from s to q_0, q_1, and q_2 in $G^{q_0}_{i-1}$, $G^{q_1}_{i-1}$, and $G^{q_2}_{i-1}$ are highlighted in red, blue, and green, respectively. The last edge of each π_j (which belongs to $E(G) \setminus E(H_{i-1})$) is drawn as a solid thin line.

Observe that both $\pi_{j'}$ and $\delta_{j'}$ belong to $\Pi(s, q_{j'}; G^{q_{j'}}_{i-1})$, hence must have the same length. However $|\delta_{j'}| = |\delta_j| + |\pi_{j'}[q_j : q_{j'}]| = |\pi_{j'}| + |\delta_j[q_{j'} : q_j]| + |\pi_{j'}[q_j : q_{j'}]|$. As consequence, $|\delta_{j'}| \neq |\pi_{j'}|$ since $\eta(\pi_{j'}[q_j : q_{j'}]) = 1$ and hence $\eta(\delta_{j'}) > \eta(\pi_{j'})$.

Define $\delta = \pi_{k-1}[q_k : q_{k-1}] \circ \pi_{k-2}[q_{k-1} : q_{k-2}] \circ \pi_{k-3}[q_{k-2} : q_{k-3}] \circ \ldots \circ \pi_0[q_1 : q_0]$. We prove by reverse induction on $j = k, \ldots, 0$ that (i) $\delta[s : q_j]$ is a shortest path from s to q_j in $G^{q_j}_{i-1}$, and (ii) all edges in $E(\delta[s : q_j]) \cap E(G)$ belong to H_{i-1}. The claim is trivially true for $j = k$ since $q_k = s$ and $\delta[s, q_k]$ is the empty path. For $j < k$, consider the path π_j and notice that $q_{j+1} = q(\pi_j)$ by definition. By induction hypothesis, we have that $\delta[s : q_{j+1}] \in \Pi(s, q_{j+1}, G^{q_{j+1}}_{i-1})$. Then, by Lemma 5, $\delta[s : q_j] = \delta[s : q_{j+1}] \circ \pi_j[q_{j+1}, q_j] \in \Pi(s, q_j, G^{q_j}_{i-1})$, which proves (i).

As far as (ii) is concerned, we only need to argue about $\pi_j[q_{j+1}, q_j]$ since $\delta[s : q_j] = \delta[s : q_{j+1}] \circ \pi_j[q_{j+1}, q_j]$ and, by induction hypothesis, we know that all edges in $E(\delta[s : q_{j+1}]) \cap E(G)$ are in $E(H_i)$. Let (u, q_j) be the last edge of $\pi_j[q_{j+1}, q_j]$ and notice that, since (u, q_j) is also the last edge of π_j, our algorithm adds (u, q_j) to H_i when q_j is considered. Moreover, by the choice of $q_{j+1} = q(\pi_j)$, the path $\pi_j[q_{j+1} : u]$ contains no edges in $E(G) \setminus E(H_{i-1})$. This means that $E(\pi_j[q_{j+1} : u]) \cap E(G)$ lies entirely in H_{i-1} and hence in H_i. This shows that all edges of $E(\pi_j[q_{j+1}, q_j]) \cap E(G)$ belong to $E(H_i)$ and proves (ii). ◀

The above lemma easily implies that H_i is a i-multipath preserver of G.

▶ **Lemma 7.** *H_i contains i edge-disjoint paths of minimum total cost from s to every $t \in V(G) \setminus \{s\}$.*

Proof. Fix a vertex $t \in V(G) \setminus \{s\}$. By induction hypothesis all edges in S^t_{i-1} belong to H_{i-1}. By Lemma 6, there is a path $\delta \in \Pi(s, t, G^t_{i-1})$ such that $E(\delta) \cap E(G) \subseteq E(H_i)$. We now use Remark 1 to build S^t_i from S^t_{i-1} and δ. It is easy to see that S^t_i must be entirely contained in H_i. ◀

The combination of Lemma 7 with the discussion on the size of H_p at the beginning of Section 3, immediately results in the following theorem.

▶ **Theorem 8.** *H_p is a single-source p-multipath preserver of size $p(n-1)$. More precisely, s has in-degree 0 in H_p while each other vertex has in-degree p.*

Algorithm 1 Computes a p-multipath preserver of a graph G and p-edge disjoint paths of minimum total cost from s to t, for every $t \in V(G) \setminus \{s\}$.

```
   Input  : A graph $G = (V(G), E(G))$, a source vertex $s \in V(G)$, $p \in \mathbb{N}^+$;
   Output : $p$ edge-disjoint paths $S_p^t$ from $s$ to $t$ of minimum total cost, $\forall t \in V(G)$;
   Output : a $p$-multipath preserver $H_p$ of $G$ with source $s$;
1  $H_1 \leftarrow$ shortest path tree of $G$ rooted at $s$;              // $H_1$ is a 1-multipath preserver
2  foreach $t \in V(G)$ do $S_1^t \leftarrow$ path from $s$ to $t$ in $H_1$;
3  for $i \leftarrow 2, \dots, p$ do                                        // Compute $H_i$ and all $S_i^t$
4      foreach $t \in V(G) \setminus \{s\}$ do
5          $H_{i-1}^t \leftarrow$ Graph obtained from $H_{i-1}$ by reversing the edges in $S_{i-1}^t$ and adding the
             edges incident to $t$ in $E(G) \setminus S_{i-1}^t$;
6          $T_{i-1}^t \leftarrow$ reverse SPT towards $t$ in $H_{i-1}^t$;
           // $\pi(u, t; T_{i-1}^t)$ is the sole path in $\Pi(u, t; T_{i-1}^t)$
       // Initialize distances and priority queue
7      $d(s) \leftarrow (0, 0)$;    $\pi_s \leftarrow$ Empty path;
8      foreach $t \in V(G) \setminus \{s\}$ do $d(t) \leftarrow (+\infty, +\infty)$;
9      $Q \leftarrow$ initialize a priority queue with values in $V(G)$ and keys $d(\cdot)$;
10     $H_i \leftarrow H_{i-1}$;
11     while $Q$ is not empty do
12         $q \leftarrow$ Extract the minimum from $Q$;
13         if $q \neq s$ then
14             $\pi_q \leftarrow \pi_{\rho(q)} \circ \pi(\rho(q), q; T_{i-1}^q)$;  // $\pi_q \in \Pi(s, q; G_{i-1}^q)$. $\rho(q)$ was set in Line 20
15             $e_q \leftarrow$ last edge of $\pi_q$;
16             $E(H_i) \leftarrow E(H_i) \cup \{e_q\}$;                   // Update the $i$-multipath preserver
17             Compute $S_i^q$ from $S_{i-1}^q$ and $\pi_q$ as explained in Remark 1
18         foreach $t \in Q$ do
               // Check whether $\pi_q \circ \pi(q, t; T_{i-1}^t)$ is shorter than $d(t)$
19             if $q \in V(T_{i-1}^t)$ and $d(q) + |\pi(q, t; T_{i-1}^t)| \prec d(t)$ then
20                 $\rho(t) \leftarrow q$;              // We found a shorter path to $t$ in $G_{i-1}^t$ (via $q$)
21                 $d(t) \leftarrow d(q) + |\pi(q, t; T_{i-1}^t)|$;                        // Relax $d(t)$
22                 Decrease the key of vertex $t$ in $Q$ to $d(t)$;
```

4 An efficient algorithm for finding p edge-disjoint shortest paths

In this section we describe an algorithm (whose pseudocode is given in Algorithm 1) running in time $O(p^2n^2 + pn^2 \log n)$ that computes: (i) p edge disjoint paths S_p^t of minimum total cost from s to t; (ii) a single-source p-multipath preserver H_p of size $p(n-1)$ (as stated in Theorem 8). Our algorithm also guarantees that each S_p^t is contained in H_p.

More precisely, the algorithm will compute along the way all single-source i-multipath presevers H_i, for $i = 1, \dots, p$, as defined in the previous section (recall that H_i has size $i(n-1)$). In this sense, the algorithm can be seen as an efficient implementation of the one described in Section 3.

The algorithm works in phases. The generic i-th phase will compute a i-multipath preserver H_i from the $(i-1)$-th multipath preserver H_{i-1} computed by the previous phase. The algorithm also maintains, for each vertex t, a solution S_i^t consisting of i edge-disjoint paths of minimum total cost from s to t. Similarly to H_i, S_i^t is computed from S_{i-1}^t during phase i.

Initially, H_1 and all S^t_1 are simply a *shortest-path tree* (SPT) of G rooted at s, and the (unique) path from s to t in H_1. In each phase $i \geq 2$, the algorithm aims to find a shortest path $\pi_t \in \Pi(s, t; G^t_{i-1})$. Since a direct computation of π_t would be too time-consuming, the idea is that of exploiting the suboptimality property of Lemma 5 to consider π_t as the composition of two subpaths π_q and $\pi_t[q : t]$, where $q = q(\pi_t)$ and π_q is a shortest path from s to q in G^q_{i-1}.

To this aim, we follow a Dijkstra-like approach (see lines 7–22). More precisely, once we have computed π_q, we attempt to extend it towards every other vertex t by concatenating π_q with a shortest path from q to t in G^t_{i-1}.

As we have discussed in Section 2, once we have π_t, we can easily compute S^t_i from S^t_{i-1} and π_t according to Remark 1. Moreover, as seen in Section 3, we compute H_i by adding to H_{i-1} all the last edges of each π_t.

There are, however, three caveats that need to be carefully handled. The first two concerns the algorithm's correctness:

- Whenever a path π_q is extended towards a vertex t, the resulting path may not necessarily exist in G^t_{i-1} (since π_q lies in G^q_{i-1} which differs from G^t_{i-1}). However, this is not an issue since, as we will prove in the following (see Lemma 10), when π_q does not exist in G^t_{i-1}, the length of the resulting path is always an upper bound to the length of π_t.
- In order for the Dijkstra-like approach to work, the vertices q need to be considered in non-decreasing order of $|\pi_q|$, and hence the shortest path from q to t in G^t_{i-1} used to extend π_q must have non-negative costs. As we will show, this is indeed the case (see Lemma 9).

The last critical aspect concerns the complexity of the algorithm: a direct computation of the needed shortest path from q to t in G^t_{i-1} would be too time-consuming.

Instead, we (pre-)compute it in a suitable sparse subgraph of G^t_{i-1}, referred as H^t_{i-1} in the pseudocode (see lines 4–6).

4.1 Proof of correctness

We prove the correctness of Algorithm 1 by induction on $i \geq 1$. In particular we will show that, at the end of phase i, the following three properties will be satisfied: (i) for $t \in V(G) \setminus \{s\}$, the edges in S^t_i induce i edge-disjoint paths of minimum total cost from s to t in G; (ii) H_i is a i-multipath preserver for G with source s; and (iii) for $t \in V(G) \setminus \{s\}$, S^t_i is entirely contained in $E(H_i)$.

The base case $i = 1$ is trivially true since H_1 is a shortest path-tree from s in G, and S^t_i is the (unique) path from s to t in H_1. We hence assume (i), (ii), and (iii) for $i - 1$ and focus on phase $i \geq 2$.

For each t, let G^t_{i-1} be the residual network obtained from G by reversing the edges of S^t_{i-1}. The rest of the proof is organized as follows: we first prove that Algorithm 1 correctly computes a shortest path $\pi_t \in \Pi(s, t; G^t_{i-1})$. Then, we will argue that this implies properties (i), (ii), and (iii).

▶ **Lemma 9.** *Let $t \in V(G)$, and consider the i-th phase of Algorithm 1. For every $q \in V(T^t_{i-1})$, we have $c(\pi(q, t; T^t_{i-1})) \geq 0$.*

Proof. Assume towards a contradiction that for some $t, q \in V(G)$, $c(\pi(q, t; T^t_{i-1})) < 0$. The cost of a shortest path π from q to t in H^t_{i-1} is $c(\pi(q, t; T^t_{i-1}))$. If $c(\pi) < 0$, it contains edges that are reversed w.r.t. G. The set of reversed edges are those belonging to $i - 1$ edge disjoint paths from t to s in G^t_{i-1}. Let (x, y) the first reversed edge traversed in π. Consider the subpath $\pi[x:t]$. It holds that $c(\pi[x:t]) \leq c(\pi)$. Consider path π' from t to x in G^t_{i-1}, that consists in only reversed edges. Thus $\pi[x:t] \circ \pi'$ is a closed walk of negative total cost in G^t_{i-1} that does not contain negative cycles [14]. ◀

▶ **Lemma 10.** *Let $t \in V(G)$, $\pi \in \Pi(s,t;G^t_{i-1})$ and consider the i-th phase of Algorithm 1. The path π_t computed by line 14 after t is extracted from Q satisfies $|\pi| \preceq |\pi_t|$.*

Proof. By contradiction, consider first extracted node t for which $\pi \not\preceq \pi_t$. Let $q = \rho(t)$ be the last node that relaxed node t during phase i. Then $\pi_t = \pi_q \circ \pi(q,t;T^t_{i-1})$.

Consider $\pi' \in \Pi(s,q;G^q_{i-1})$, by hypothesis $\pi' \preceq \pi_q$. There are two cases: (i) π' exists in G^t_{i-1}, (ii) π' does not exists in G^t_{i-1}. In the first case, $\pi' \circ \pi(q,t;T^t_{i-1})$ exists in G^t_{i-1} then, $\pi \preceq \pi' \circ \pi(q,t;T^t_{i-1}) \preceq \pi_q \circ \pi(q,t;T^t_{i-1}) = \pi_t$. In the second case by Lemma 4, there exists a path π'' from s to q in G^t_{i-1} such that $\pi'' \prec \pi'$. Observe that $\pi'' \circ \pi(q,t;T^t_{i-1})$ is an existing path in G^t_{i-1}. Then, $\pi \preceq \pi'' \circ \pi(q,t;T^t_{i-1}) \prec \pi' \circ \pi(q,t;T^t_{i-1}) \preceq \pi_q \circ \pi(q,t;T^t_{i-1}) = \pi_t$. ◀

Since, for each node t, the value of $d(t)$ is initialized to $(+\infty,+\infty)$ and it is only decreased during the while loop, the above lemma implies that $d(t)$ is an upper bound on the value $|\pi|$, with $\pi \in \Pi(s,t;G^t_{i-1})$.

▶ **Lemma 11.** *Let $t \in V(G)$, $\pi \in \Pi(s,t;G^t_{i-1})$ and consider the i-th phase of Algorithm 1. The subpath $\pi[q(\pi):t]$ is entirely contained in H^t_{i-1}.*

Proof. Recall that H^t_{i-1} is defined as the graph obtained from H_{i-1} where the edges in S^t_{i-1} are reversed and contains all the edges $E(G) \setminus S^t_{i-1}$ entering in t.

By definition, $\pi[q(\pi):t]$ consists in a sequence of edges (possibly reversed) in H_{i-1} and one edge in $E(G) \setminus E(H_{i-1})$ entering in t. Since by inductive hypothesis $S^t_{i-1} \subseteq E(H_{i-1})$ and S^t_{i-1} is the set of reversed edges in G^t_{i-1}, then $\pi[q(\pi):t]$ exists in H^t_{i-1}. ◀

▶ **Lemma 12.** *Let $t \in V(G)$, $\pi \in \Pi(s,t;G^t_{i-1})$ and consider the i-th phase of Algorithm 1. The path π_t computed by line 14 after t is extracted from Q satisfies $|\pi_t| \preceq |\pi|$.*

Proof. By contradiction, take the first extracted node t for which $\pi_t \not\preceq \pi$. For simplicity let $q = q(\pi)$.

By the suboptimality property (Lemma 5), we have that $\pi = \pi' \circ \pi[q:t]$, where $\pi' \in \Pi(s,q;G^q_{i-1})$. By Lemma 11 $\pi[q:t]$ exists in H^t_{i-1}. Notice that since $H^t_{i-1} \subseteq G^t_{i-1}$, then $|\pi[q:t]| = |\pi(q,t;T^t_{i-1})|$ and by Lemma 9, $c(\pi[q:t]) \geq 0$. Moreover $\pi[q:t]$ contains one edge in $E(G) \setminus E(H_{i-1})$ thus $\pi' \prec \pi$. By hypothesis, $\pi_q \preceq \pi'$ and by Lemma 10 $\pi \preceq \pi_t$, hence $\pi_q \prec \pi_t$ and node q is extracted before t. Line 19 of Algorithm 1 ensures that $\pi_t \preceq \pi_q \circ \pi(q,t;T^t_{i-1}) \preceq \pi' \circ \pi(q,t;T^t_{i-1}) = \pi$. ◀

▶ **Lemma 13.** *Let $t \in V(G)$, $\pi \in \Pi(s,t;G^t_{i-1})$ and consider the i-th phase of Algorithm 1. The path π_t computed by line 14 after t is extracted from Q is entirely contained in G^t_{i-1}.*

Proof. By contradiction, take first extracted node t for which π_t does not exists in G^t_{i-1}. Let q be the last node that performed a relaxation for t, we have that $\pi_t = \pi_q \circ \pi(q,t;T^t_{i-1})$. Since $\pi(q,t;T^t_{i-1})$ exists in G^t_{i-1} then π_q does not. By hypothesis π_q exists in G^q_{i-1} and by Lemma 4 there exists a path π' in G^t_{i-1} such that $\pi' \prec \pi_q$. The path $\pi' \circ \pi(q,t;T^t_{i-1})$ gives us an existing path in G^t_{i-1}, such that $\pi \preceq \pi' \circ \pi(q,t;T^t_{i-1}) \prec \pi_t$, where $\pi \in \Pi(s,t;G^t_{i-1})$. This contradicts Lemma 12 for which $\pi_t \preceq \pi$. ◀

We are now ready to establish the correctness of the algorithm as summarized by the following lemma.

▶ **Lemma 14.** *For all $i = 1, \dots, p$ and $t \in V(G) \setminus \{s\}$, Algorithm 1 computes a single-source i-multipath preserver H_i of G and a set S_i^t, with $S_i^t \subseteq E(H)$, inducing i edge-disjoint paths of minimum total cost from s to t in G.*

Proof. Consider a vertex $t \in V(G) \setminus \{s\}$. By Lemma 12 and Lemma 13, we have that at the end of phase i, $\pi_t \in \Pi(s, t; G_{i-1}^t)$. Then, by the inductive hypothesis and by Remark 1, the set S_i^t contains i-edge disjoint paths from s to t of minimum total cost in G, as desired. Moreover, since phase i constructs H_i by augmenting H_{i-1} with the last edge of every π_t, as shown in Section 3, we have that H_i is a single-source i-multipath preserver of G.

It remains to prove that, at the end of phase i of Algorithm 1, all edges in S_i^t are in H_i.

For any vertex t, let π_t be the path computed in line 14 of Algorithm 1 during phase i. Notice that, by construction (see Remark 1), each edge in S_i^t belongs to at least one of S_{i-1}^t and $E(\pi_t) \cap E(G)$. Since we already know that $S_{i-1}^t \subseteq E(H_{i-1}) \subseteq E(H_i)$, we only need to show that $E(\pi_t) \cap E(G) \subseteq E(H_i)$.

We consider all paths π_t computed by Algorithm 1 during phase i, and we prove the above property by induction on the number of edges ℓ of π_t.

The base case $\ell = 0$ is trivially true since any such path contains no edges. Consider now a path π_t with $\ell \geq 1$ edges. Since $t \neq s$, π_t has been computed in line 14 as the concatenation of a path $\pi_{\rho(t)}$ with $\pi' = \pi(\rho(t), t; T_{i-1}^t)$. The path π' is entirely contained in $T_{i-1}^t \subseteq H_{i-1}^t$. By construction of H_{i-1}^t, the only edges of G that are in $E(H_{i-1}^t) \setminus E(H_i)$ enter t. Let $e_t = (u, t)$ be the last edge π' (this edge always exists since $\rho(t) \neq t$). By the above observation we have that $E(\pi'[s : u]) \cap E(G) \subseteq E(H_{i-1}) \subseteq E(H_i)$, while e_t is added to H_i by line 16. Finally, $\pi_{\rho(t)}$ satisfies $E(\pi_{\rho(t)}) \cap E(G) \subseteq E(H_i)$ by inductive hypothesis, since $\pi_{\rho(t)}$ has less edges than π_t. ◀

4.2 Analysis of the computational complexity

In order to bound the time complexity of our algorithm we first argue about how, during phase i of Algorithm 1, it is possible to implement Line 6 in time $O(in + n \log n)$.

For any fixed phase i of the algorithm, and for any target vertex t, Line 6 computes a (reverse) shortest path tree towards t in H_{i-1}^t. As the edge costs in H_{i-1}^t can be negative, a naive implementation using the Bellman-Ford algorithm would require $\Theta(in^2)$ time (since H_{i-1}^t has size $\Theta(in)$). Consequently, the overall time needed to compute all trees T_{i-1}^t, for every i and every t, would be $\Theta(p^2 n^3)$.

To reduce the time complexity of this step, we use a technique similar to the one employed in the successive shortest path algorithm: we re-weight the edges of H_{i-1}^t so that (i) shortest paths are preserved, and (ii) all edge costs are non-negative. Then, after such a re-weighting, a SPT towards t in H_{i-1}^t can be found in $O(in + n \log n)$ time using Dijkstra's algorithm.

We will employ a well-known re-weighting scheme in which the edge costs are completely determined by some function $h : V \to \mathbb{R}$ (see, e.g., [12, Ch 25.3]). Given h, the *new cost* $c'(u, v)$ of an edge (u, v) is defined as $c(u, v) + h(u) - h(v)$. Notice that the cost of any path π from x to y w.r.t. c' is exactly $c(\pi) + h(x) - h(y)$, thus the set of shortest paths w.r.t. c' coincides with the corresponding set w.r.t. c. Therefore, the above re-weighting scheme immediately satisfies (i), and hence we will only need to argue about (ii).

Suppose that, at the beginning of phase i (where i ranges from 2 to p), we already know a re-weighting function h_{i-2}^t such that the graph G_{i-2}^t re-weighted according to h_{i-2}^t has no negative-cost edges.[5] We will show how to use h_{i-2}^t to obtain a new re-weighting function

[5] Observe that, in the first phase $i = 2$, such a function h_0^t is trivially known. Indeed, since the edge costs of G are already non-negative we can simply choose $h_0^t(v) = 0$ for each $v \in V(G)$.

h^t_{i-1} such that the graph G^t_{i-1} re-weighted according to h^t_{i-1} has no negative-cost edges. Since H^t_{i-1} is a subgraph of G^t_{i-1}, h^t_{i-1} also satisfies (ii). The re-weighting induced by h^t_{i-1} can then be immediately used to implement Line 6 of the algorithm in time $O(in + n \log n)$ using Dijkstra's algorithm.

Let $\bar{H}$ be a graph obtained from H_{i-1} by reversing the edges in S^t_{i-2} (i.e., $i-2$ edge-disjoint paths of minimum total cost from s to t), and notice that $\bar{H}$ is a subgraph of G^t_{i-2}. We compute all the distances from s to the vertices in $\bar{H}$ using Dijkstra algorithm (where edges are re-weighted w.r.t. h^t_{i-2}) and we let $h(v)$ be the distance from s to v. Finally, for each $v \in V(G)$, we define $h^t_{i-1}(v) = h^t_{i-2}(v) + h(v)$. In the following we prove that all edge costs are non-negative once G^t_{i-1} is re-weighted according to h^t_{i-1}. We start with a technical lemma showing that the distances from s in $\bar{H}$ and G^t_{i-2} coincide.

▶ **Lemma 15.** *Let T be a shortest path tree rooted at s of $\bar{H}$ then, T is also a shortest path tree rooted at s of G^t_{i-2}.*

Proof. By contradiction, if T is not a shortest path tree of G^t_{i-2}, it is because there exists some node v in G^t_{i-2} for which every path $\pi \in \Pi(s, v; G^t_{i-2})$ contains some edge from $E(G) \setminus E(\bar{H})$. Fix a $\pi \in \Pi(s, v; G^t_{i-2})$ and assume w.l.o.g. that π contains only one edge from $E(G) \setminus E(\bar{H})$ and that this edge enters in v.

We now show that $\pi(s, v; T) \preceq \pi$. Consider $\pi_v \in \Pi(s, v; G^v_{i-2})$ computed by Algorithm 1 during phase $i-1$, and observe that for each $(u, v) \in \pi_v$ either $(u, v) \in E(H_{i-1})$ or $(v, u) \in E(H_{i-1})$. By Lemma 4, either π_v exists in G^t_{i-2} or there exists a path π'_v in G^t_{i-2}, obtained from π_v by substituting a subpath in π_v with a path containing only edges from $\Delta(t, v)$ w.r.t. G^t_{i-2} and G^v_{i-2} and such that $\pi'_v \prec \pi_v$. Let π' be the existing path in G^t_{i-2} between π'_v and π_v. By construction, π' is a path that consists only in edges from $\bar{H}$, thus $\pi(s, v; T) \preceq \pi'$.

To conclude the proof, we need to show that π exists in G^v_{i-2}. Similarly to the proof of Lemma 3, if π does not exists in G^v_{i-2}, it traverses at least one edge in $\Delta(t, v)$ w.r.t. G^t_{i-2} and G^v_{i-2}. Let (x, y) be the last edge traversed by π that belongs in $\Delta(t, v)$. Let δ be a path from t to y that traverses (x, y) in the subgraph of G^t_{i-2} induced by the edges in $\Delta(t, v)$.

Since, by definition of π, the edge e of π entering in v is not in $\bar{H}$, we have $e \notin \Delta(t, v)$. Then, the subpath $\pi[y : v]$ of π is not empty and, by our choice of (x, y) does not traverse any edge in $\Delta(t, v)$.

By the suboptimality property of shortest paths, $\pi[y : v] \preceq \delta[y : v]$ and hence $c(\pi[y : v]) \leq c(\delta[y : v])$. If $c(\pi[y : v]) < c(\delta[y : v])$, we can replace $\delta[y : v]$ with $\pi[y : v]$ in δ to obtain a path δ' from t to v in G^t_{i-2} with $c(\delta') < c(\delta)$. This contradicts Lemma 2 since it implies the existence of $i-2$ edge-disjoint paths from t to v in G^t_{i-2} with a total cost smaller than $c(\Delta(t, v))$.

If $c(\pi[y : v]) = c(\delta[y : v])$, we can replace $\pi[y : v]$ with $\delta[y : v]$ in π to obtain a path π'' from s to v in G^t_{i-2} satisfying $c(\pi'') = c(\pi)$. Since all edges (or their reverse) of $E(\delta[y : v]) \subseteq \Delta(t, v)$ are in H_{i-2}, $\pi[y : v]$ contains more edges in $E(G) \setminus E(H_{i-2})$ than $\delta[y : v]$, thus $\pi'' \prec \pi$. This is a contradiction since, by the suboptimality property of shortest paths and by our choice of $\pi \in \Pi(s, v; G^t_{i-2})$, π must be a shortest path from s to v in G^t_{i-2} w.r.t. $\preceq$.

Then knowing that π exists also in G^v_{i-2}, it holds that $\Pi(s, v; T) \preceq \pi' \preceq \pi$. ◀

▶ **Lemma 16.** *For any $(u, v) \in E(G^t_{i-1})$, we have $c(u, v) + h^t_{i-1}(u) - h^t_{i-1}(v) \geq 0$.*

Proof. Let $c'(u, v)$ denote the cost of edge (u, v) in G^t_{i-2}, when the graph is re-weighted according to h^t_{i-2}. Notice that, by hypothesis, $c'(u, v)$ is always non-negative. Recall that $h(v)$ is the distance from s to v in $\bar{H}$ w.r.t. c' and that, by Lemma 15, $h(v)$ is also the distance from s to v in G^t_{i-2} w.r.t. the cost function c'.

Thus we have that, for each edge $(u,v) \in E(G^t_{i-2})$, $h(u) + c'(u,v) \geq h(v)$ implying that $c(u,v) + h^t_{i-1}(u) - h^t_{i-1}(v) = c'(u,v) + h(u) - h(v) \geq 0$. In particular, if $(u,v) \in E(G^t_{i-2})$ belongs to a shortest path (w.r.t. c') from s to v in G^t_{i-2} then, $h(u) + c'(u,v) = h(v)$ and $c(u,v) + h^t_{i-1}(u) - h^t_{i-1}(v) = c'(u,v) + h(u) - h(v) = 0$.

By definition, G^t_{i-1} is obtained from G^t_{i-2} by reversing $\pi \in \Pi(s,t; G^t_{i-2})$.

For each $(u,v) \in E(G^t_{i-1}) \cap E(G^t_{i-2}), c(u,v) + h^t_{i-1}(u) - h^t_{i-1}(v) \geq 0$ and for each $(u,v) \in E(G^t_{i-1}) \setminus E(G^t_{i-2})$ we have that $(v,u) \in \pi$ then, $c(u,v) + h^t_{i-1}(u) - h^t_{i-1}(v) = -(c(v,u) - h^t_{i-1}(u) + h^t_{i-1}(v)) = 0$. ◀

We conclude by observing that $\bar{H}$ can be computed in $O(in)$ and therefore the overall running time required to compute T^t_i is $O(in + n \log n)$, as claimed. The overall running time of Algorithm 1 is $O(p^2n^2 + pn^2 \log n)$. The next theorem follows from Theorem 8, Lemma 14, and the above discussion.

▶ **Theorem 17.** *Algorithm 1 solves both single-source p-multipath preserver problem and shortest p edge-disjoint paths problem in $O(p^2n^2 + pn^2 \log n)$. Moreover, the size of the computed preserver is equal to $p(n-1)$, which is optimal.*

Proof. We can ignore lines 1 and 2 since they require time $O(n^2)$. We therefore focus on an iteration $i \geq 2$ of the outer loop (i.e., on phase i).

The discussion in Section 4.2 shows that the loop at lines 4–6 requires time $O(in^2 + n^2 \log n) = O(pn^2 + n^2 \log n)$. Observe that line 14 can be implemented in time proportional to the number of edges of π_q, which is at most $n-1$, and that line 17 requires time at most $O(|S^q_{i-1}| + n) = O(pn)$. We implement the priority queue Q using a data structure that supports decrease-key operations in constant-time (e.g., an array). Since we perform $O(n)$ extract-min operations, and $O(n^2)$ decrease-key operations, we have that the loop at lines 7–22 requires time $O(in^2) = O(pn^2)$.

Thus, the overall time complexity of the algorithm is $O(p^2n^2 + pn^2 \log n)$. ◀

References

1 Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen G. Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Comput. Sci. Rev.*, 37:100253, 2020. doi:10.1016/j.cosrev.2020.100253.

2 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications.* Prentice Hall, 1993.

3 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discret. Comput. Geom.*, 9:81–100, 1993. doi:10.1007/BF02189308.

4 Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate single-source fault tolerant shortest path. *ACM Trans. Algorithms*, 16(4):44:1–44:22, 2020. doi:10.1145/3397532.

5 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault-tolerant subgraph for single-source reachability: General and optimal. *SIAM J. Comput.*, 47(1):80–95, 2018. doi:10.1137/16M1087643.

6 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Multiple-edge-fault-tolerant approximate shortest-path trees. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.18.

7 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Fault-tolerant approximate shortest-path trees. *Algorithmica*, 80(12):3437–3460, 2018. doi:10.1007/s00453-017-0396-z.

8 Greg Bodwin. Linear size distance preservers. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 600–615. SIAM, 2017. doi:10.1137/1.9781611974782.39.

9 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 73:1–73:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.73.

10 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 855–872. SIAM, 2016. doi:10.1137/1.9781611974331.ch61.

11 Shiri Chechik, Quentin Godfroy, and David Peleg. Multipath spanners via fault-tolerant spanners. In Guy Even and Dror Rawitz, editors, *Design and Analysis of Algorithms - First Mediterranean Conference on Algorithms, MedAlg 2012, Kibbutz Ein Gedi, Israel, December 3-5, 2012. Proceedings*, volume 7659 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2012. doi:10.1007/978-3-642-34862-4_8.

12 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

13 Paul Erdős. Extremal problems in graph theory. In *Theory of Graphs and its Applications.* Academic Press, New York, 1965.

14 Jeff Erickson. *Algorithms.* independent, 2019. Chapter G: *Minimum-Cost Flows* in the *Extended Dance Remix* available online. URL: http://jeffe.cs.illinois.edu/teaching/algorithms/.

15 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

16 Harold N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. Syst. Sci.*, 50(2):259–273, 1995. doi:10.1006/jcss.1995.1022.

17 Cyril Gavoille, Quentin Godfroy, and Laurent Viennot. Multipath spanners. In Boaz Patt-Shamir and Tínaz Ekim, editors, *Structural Information and Communication Complexity, 17th International Colloquium, SIROCCO 2010, Sirince, Turkey, June 7-11, 2010. Proceedings*, volume 6058 of *Lecture Notes in Computer Science*, pages 211–223. Springer, 2010. doi:10.1007/978-3-642-13284-1_17.

18 Cyril Gavoille, Quentin Godfroy, and Laurent Viennot. Node-disjoint multipath spanners and their relationship with fault-tolerant spanners. In Antonio Fernández Anta, Giuseppe Lipari, and Matthieu Roy, editors, *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, volume 7109 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2011. doi:10.1007/978-3-642-25873-2_11.

19 Manoj Gupta and Shahbaz Khan. Multiple source dual fault tolerant BFS trees. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 127:1–127:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.127.

20 Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. A brief note on single source fault tolerant reachability. *CoRR*, abs/1904.08150, 2019. arXiv:1904.08150.

21 Merav Parter. Dual failure resilient BFS structure. In Chryssis Georgiou and Paul G. Spirakis, editors, *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 481–490. ACM, 2015. doi:10.1145/2767386.2767408.

22 Merav Parter and David Peleg. Sparse fault-tolerant BFS structures. *ACM Trans. Algorithms*, 13(1):11:1–11:24, 2016. doi:10.1145/2976741.

23 Merav Parter and David Peleg. Fault-tolerant approximate BFS structures. *ACM Trans. Algorithms*, 14(1):10:1–10:15, 2018. doi:10.1145/3022730.

24 David Peleg and Alejandro A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

25 J. W. Suurballe and Robert Endre Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984. doi:10.1002/net.3230140209.

A Extensions and variants

In this section we show how to extend all our results to more general versions of the two problems in which the input graph G is not necessarily p-edge-outconnected from s. More precisely, we denote by $\lambda(t)$ the maximum number of edge-disjoint paths from s to t in G. We want to find, for each vertex $t \in V(G) \setminus \{s\}$, $\sigma(t) := \min\{\lambda(t), p\}$ edge-disjoint paths of minimum total cost from s to t in G.

Moreover, we show how to use Algorithm 1 to approximate the problem of computing a set of p edge-disjoint paths where the cost of the path with maximum cost is minimized. We show that the approximation factor achieved by our algorithm is optimal.

Finally, for the sake of completeness, we also describe the graph transformation already discussed in [25] if we are interested in finding paths that are vertex-disjoint rather than edge-disjoint.

Extensions to general versions of our problems. W.l.o.g., we can assume that $p < n$ as G contains at most $n-1$ edge-disjoint paths from s to any vertex $t \in V(G) \setminus \{s\}$.

We transform the input graph G into another graph G' that is p-edge-outconnected from s. To construct G', we take a copy of G and augment it by adding a complete directed graph C on p new "dummy vertices" $v_1, v_2, \ldots, v_p$, all edges in $\{s\} \times \{v_1, v_2, \ldots, v_p\}$, and all edges in $\{v_1, v_2, \ldots, v_p\} \times (V(G) \setminus \{s\})$, where the cost of all the new edges is some large value $M > c(E(G))$. We observe that each edge of cost M is incident to at least one dummy vertex. Furthermore, there are p edge-disjoint paths from s to any other vertex of the vertex of the graph, so H' is p-edge-outconnected from s. As $p < n$, the graph G' still contains $O(n)$ vertices. We run Algorithm 1 on G' to compute all the sets S^t_i, for each $t \in V(G) \setminus s$ and $i \leq p$ (Lemma 14) in $O(p^2n^2 + pn \log n)$ time. The solution to our problem for t is given by $S^t_{\sigma(t)}$, where we can find the value of $\sigma(t)$ as the largest index i for which $c(S^t_i) < M$.

Concerning the problem of finding a subgraph H of G such that $S^t_{\sigma(t)} \subseteq E(H)$ for every $t \in V(G) \setminus \{s\}$, we first compute a single-source p-multipath preserver H' of G' in $O(p^2n^2 + pn \log n)$ time using Algorithm 1. The graph H is obtained from H' by deleting all the dummy vertices and, consequently, all the edges (each of cost M) that are incident to the dummy vertices. We observe that H, being a subgraph of G, does not contain edges of cost M.

▶ **Theorem 18.** *For every $t \in V(G) \setminus \{s\}$, $S^t_{\sigma(t)} \subseteq E(H)$. Moreover, the size of H is equal to $\sum_{t \in V(G) \setminus \{s\}} \sigma(t)$, which is optimal.*

Proof. By the algorithmic construction of the single-source p-multipath preserver, we have that $H' = H'_p$ contains H'_i, for every $i \leq p$. Since the edges of $S^t_{\sigma(t)}$ are also edges of $H'_{\sigma(t)}$ (Lemma 14), it follows that $S^t_{\sigma(t)} \subseteq E(H'_p)$, and thus $S^t_{\sigma(t)} \subseteq E(H)$, as $S^t_{\sigma(t)}$ has no edge of cost M.

The lower bound of $\sum_{t \in V(G) \setminus \{s\}} \sigma(t)$ on the size of any feasible solution to the problem comes from the fact that the in-degree of each vertex t must be at least $\sigma(t)$.

We now prove that the size of H matches the lower bound by showing that the in-degree of each vertex $t \in V(G) \setminus \{s\}$ equals $\sigma(t)$. Using the fact that the in-degree of t in H' is exactly equal to p (Theorem 17), it is enough to show that there are $p - \sigma(t)$ edges of cost M that are entering t in H'.

Consider the solution S_p^t that contains p edge-disjoint paths $\pi_1, \ldots, \pi_t$ from s to t in G' of minimum total cost. W.l.o.g., we assume that $c(\pi_1) \leq \cdots \leq c(\pi_p)$. We claim that for each i with $\sigma(t) < i \leq p$, π_i enters t with an edge of cost M. To show this it is enough to observe two things. From the one hand, $c(\pi_i)$ must have a cost of at least M as otherwise we would have $\sigma(t) + 1$ edge-disjoint paths from s to t in G of total cost of at most $c(E(G)) < M$, thus contradicting the assumption that there are at most $\sigma(t) = \lambda(t)$ edge-disjoint paths from s to t in G. On the other hand, every path from s to t in G' of cost of at least M has a cost that is actually lower bounded by $2M$. This is because any such path must pass through a dummy vertex which has only edges of cost M incident to it. As a consequence, $c(\pi_i) \geq 2M$ for every $\sigma(t) < i \leq p$.

To complete the proof, it is enough to notice that each path of cost equal to $2M$ from s to t passes through a single dummy vertex and enters in t with an edge of cost M. As there are p dummy vertices, there are also p edge-disjoint paths from s to t of cost $2M$ each. This implies that each path π_i from s to t of cost strictly larger than $2M$ can be replaced by a path of cost exactly equal to $2M$ using shortcuts (i.e., the direct edge from the first dummy vertex traversed in π to t). If we do this simultaneously for all the paths $\pi_1, \ldots, \pi_p$ of total cost strictly larger than $2M$, we obtain a new set of paths $\pi'_1, \ldots, \pi'_p$ that are still pairwise edge-disjoint and such that $\sum_{i=1}^{p} c(\pi'_i) < \sum_{i=1}^{p} c(\pi_i)$. Therefore, by the optimality of S_p^t, $c(\pi_i) = 2M$ for every $\sigma(t) < i \leq p$. As a consequence, each π_i, with $\sigma(t) < i \leq p$, enters in t with an edge of cost M. Therefore, $p - \sigma(t)$ edges out the p edges entering t in H' are of cost M each. Hence, the degree of t in H is equal to $\sigma(t)$. ◀

Computing edge-disjoint paths with minimum maximum cost. We now consider a variant of the shortest p edge-disjoint paths problem in which we have a different objective function: we want to find, for a given source vertex s and every $t \in V(G) \setminus \{s\}$, p edge-disjoint paths from s to t such that the cost of the path with maximum cost is minimized. More formally, we want to find, for each $t \in V(G) \setminus \{s\}$, a set $\bar{S}_p^t$ of p edge-disjoint paths from s to t that minimize $\max_{\pi \in \bar{S}_p^t} c(\pi)$. We call this problem the minimum bottleneck p edge-disjoint paths problem.

We observe that, for each t, the paths induced by a solution S_p^t for the shortest p edge-disjoint path problem guarantees an approximation factor of p. Indeed, $c(S_p^t) \leq \sum_{\pi \in \bar{S}_p^t} c(\pi) \leq p \cdot \max_{\pi \in \bar{S}_p^t} c(\pi)$. In the next theorem we show that this approximation factor is optimal, unless P = NP.

▶ **Theorem 19.** *There is no polynomial-time algorithm that approximates the minimum bottleneck p edge-disjoint paths problem to within a factor smaller that p, unless* P = NP.

Proof. We prove the statement for a given pair of nodes s and t. We reduce from the 2 directed paths problem (2DP): Given a directed graph $G = (V(G), E(G))$ and four vertices $s_1, t_1, s_2, t_2 \in V(G)$, decide if there exist two edge disjoint paths, one from s_1 to t_1 and one from s_2 to t_2. The 2DP problem is NP-Complete [15].

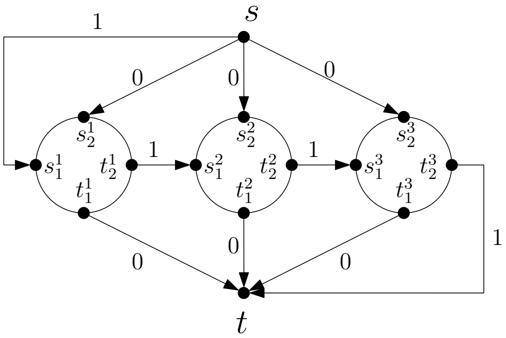


Figure 7 Reduction used in Theorem 19.

Starting from the input graph G of 2DP, we define a graph G' which consists in $p-1$ copies $G_1, \dots G_{p-1}$ of G and two new vertices s and t and we set the cost of each edge in every copy to 0. We denote by s_j^i and t_j^i, the nodes s_j and t_j in the i-th copy of G, respectively, for each $1 \leq i \leq p-1$ and $j = 1, 2$. Node s has $p-1$ edges of cost 0 toward nodes s_2^i, for each $1 \leq i \leq p-1$, and one edge of cost 1 toward node s_1^1. Node t has $p-1$ edges of cost 0 from nodes t_1^i, for each $1 \leq i \leq p-1$, and one edge of cost 1 from t_2^{p-1}. Moreover, there is an edge (t_2^i, s_1^{i+1}) of cost 1, for each $1 \leq i \leq p-2$, see Figure 7 for an illustration.

We first show that, if in G there are two edge-disjoint paths, one from s_1 to t_1 and one from s_2 to t_2, then in G' there are p edge-disjoint paths from s to t each of them with cost 1. For each $1 \leq i \leq p-1$, let us denote by π_1^i and π_2^i the two disjoint paths in G_i from s_1^i to t_1^i and from s_2^i to t_2^i, respectively. The first of the p edge-disjoint paths from s to t in G' starts from s, goes to s_1^1, follows path π_1^1 and then goes from t_1^1 to t. The total cost of this path is 1. A second path starts from s goes to s_2^{p-1}, follows path π_2^{p-1} from s_2^{p-1} to t_2^{p-1} in G_{p-1} and then goes from t_2^{p-1} to t. The total cost of this path is 1. The remaining $p-2$ are constructed in this way: Each path i, with $1 \leq i \leq p-2$, starts from s goes to s_2^i, follows path π_2^i from s_2^i to t_2^i in G_i and then crosses edge (t_2^i, s_1^{i+1}). In G_{i+1}, it follows path π_1^{i+1} from s_1^{i+1} to t_1^{i+1} and finally crosses edge (t_1^{i+1}, t). The cost of each of these paths is 1. By construction these p paths are edge-disjoint.

Now we show that, if in G there are not two edge disjoint paths, one from s_1 to t_1 and one from s_2 to t_2, then in G' any p edge disjoint paths from s to t contain a path of cost p. We can assume w.l.o.g. that there are 2 edge-disjoint paths in G, one from s_1 to t_2 and one from s_2 to t_1. In G' there is only one possible set of p edge-disjoint paths, which is made of $p-1$ paths of cost 0 and one path of cost p. The first $p-1$ paths are composed as follows: for $1 \leq i \leq p-1$, each of these paths starts from s and goes to node s_2^i in G_i through edge (s, s_2^i), it follows a path π_1^i from s_2^i to t_1^i (in G_i) disjoint from a path π_2^i between (s_1^i, t_2^i) (in G_i), and then reaches t by edge (t_1^i, t). The last path starts from s and by crossing edge (s, s_1^1) of cost 1, follows π_2^i to reach node t_2^1. At this point, it keeps moving along all copies G_i of G by using edges (t_2^i, s_1^{i+1}) of cost 1 and by using path π_2^i to reach t_2^i from s_1^i. Finally, the last edge crossed is (t_2^{p-1}, t). The total cost of this path is p.

It follows that an algorithm that approximates the minimum bottleneck p edge-disjoint paths problem to within a factor smaller that p can be used to solve 2DP. ◀

Vertex-disjoint paths. As also shown by Suurballe and Tarjan [25], all our results can be extended to the case in which the p paths of minimum total cost from s to $t \in V(G) \setminus \{s\}$ must be pairwise vertex-disjoint via the following linear time reduction. We construct a graph G' by replacing each vertex $v \in V(G)$ with a pair of vertices v^-, v^+ that are connected through an edge (v^-, v^+) with cost 0, and by adding to $E(G')$ an edge (u^+, v^-) of cost $c(u, v)$ for each edge $(u, v) \in E(G)$. We observe that G' still has $O(n)$ vertices. Although G' may not be p-edge-outconnected from s (the in-degree of each vertex v^+ is equal to 1), we can solve the problem in $O(p^2n^2 + pn \log n)$ time using Algorithm 1, via the graph transformation that adds p dummy vertices, as described in the previous paragraph.

Undirected graphs. Our results extend to the case in which the input graph G in undirected. We start by transforming G in a directed graph G': for every edge $\{u, v\} \in E(G)$, G' contains a pair of directed edges $(u, v), (v, u)$ of the same cost of $\{u, v\}$. We then invoke Algorithm 1, and transform the computed solutions for G' into solutions for G in linear time as follows.

For every $t \in V(G) \setminus \{s\}$, let S^t be a set of edges from G' that induce p edge-disjoint paths of minimum total cost from s to t in G'. We can assume w.l.o.g. that for each $(u, v) \in S^t$, $(v, u) \notin S^t$. Indeed, if $(u, v), (v, u) \in S^t$, then the set obtained by removing both (u, v) and (v, u) from S^t is still feasible.

Clearly, the solution obtained from S^t by replacing each directed edge (u, v) with the undirected edge $\{u, v\}$ still induces p edge-disjoint paths from s to t in G and has the same cost as S^t. Moreover, any set of p edge-disjoint paths from s to t in G, can also be transformed into a corresponding solution in G' by suitably orienting the traversed edges. This doesn't affect the solution's cost.

Observe that, by Lemma 14, all sets S^t are contained in the p-multipath preserver H_p of G' computed by Algorithm 1 and, by Theorem 8, the undirected version of H_p is a p-multipath preserver of G of size at most $p(n-1)$.

A 10-Approximation of the $\frac{\pi}{2}$-MST

Ahmad Biniaz ✉
School of Computer Science, University of Windsor, Canada

Majid Daliri ✉
School of Electrical and Computer Engineering, University of Tehran, Iran

Amir Hossein Moradpour ✉
School of Electrical and Computer Engineering, University of Tehran, Iran

Abstract

Bounded-angle spanning trees of points in the plane have received considerable attention in the context of wireless networks with directional antennas. For a point set P in the plane and an angle α, an α-spanning tree (α-ST) is a spanning tree of the complete Euclidean graph on P with the property that all edges incident to each point $p \in P$ lie in a wedge of angle α centered at p. The α-minimum spanning tree (α-MST) problem asks for an α-ST of minimum total edge length. The seminal work of Anscher and Katz (ICALP 2014) shows the NP-hardness of the α-MST problem for $\alpha = \frac{2\pi}{3}, \pi$ and presents approximation algorithms for $\alpha = \frac{\pi}{2}, \frac{2\pi}{3}, \pi$.

In this paper we study the α-MST problem for $\alpha = \frac{\pi}{2}$ which is also known to be NP-hard. We present a 10-approximation algorithm for this problem. This improves the previous best known approximation ratio of 16.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Approximation algorithms analysis

Keywords and phrases Euclidean spanning trees, approximation algorithms, bounded-angle visibility

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.13

Funding *Ahmad Biniaz*: supported by NSERC.

1 Introduction

Wireless antennas in a wireless network can be modeled by disks in the plane, where the centers of the disks represent locations of antennas and their radii represent transmission ranges of antennas. Two antennas can communicate if they are in each other's transmission range. In this model antennas are assumed to be omni-directional which can transmit and receive signals in 360 degrees. Replacing omni-directional antennas with *directional antennas* has received considerable attention in recent years, see for example [1, 3, 6, 8, 9, 10, 11, 13, 14, 21]. Directional antennas can transmit and receive signals only in a circular wedge with some bounded-angle α. As noted in [4, 21, 23] such a bounded-angle communication is more secure, requires lower transmission range, and causes less interference. In this model two antennas can communicate if each one is inside the other's wedge. This model is known as *symmetric communication network* [4, 5, 23].

The network connectivity is a common problem in designing networks with directional antennas. Aschner and Katz [3] formulated this problem in terms of an *α-spanning tree* (α-ST). For a point set P in the plane and an angle α, an α-ST of P is a spanning tree of the complete Euclidean graph on P such that all edges incident to each point $p \in P$ lie in a wedge of angle α centered at p (see Figure 1). It is known that an α-ST always exists when $\alpha \geqslant \frac{\pi}{3}$ (see e.g. [1, 2, 11]) while it may not exists when $\alpha < \frac{\pi}{3}$, for example if P consists of the three vertices of an equilateral triangle.

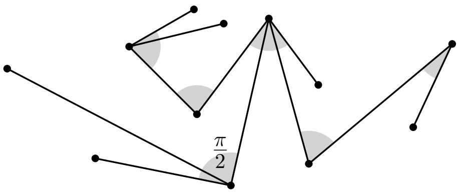


Figure 1 A $\frac{\pi}{2}$-spanning tree.

The minimum spanning tree (MST) is the shortest connected network for omni-directional antennas. For directional antennas, the shortest connected network is called the α-*minimum spanning tree* (α-MST) which is an α-ST of P with minimum total edge length. Although one can compute an MST of n points in the plane optimally in $O(n \log n)$ time, it is not clear how to efficiently compute an α-MST. Aschner and Katz [3] proved that the α-MST problem is NP-hard for $\alpha = \frac{2\pi}{3}$ and $\alpha = \pi$. They also presented approximation algorithms with ratios 16, 6, and 2 for angles $\alpha = \frac{\pi}{2}$, $\alpha = \frac{2\pi}{3}$ and $\alpha = \pi$, respectively. The approximation ratio 6 for the $\frac{2\pi}{3}$-MST has been successively improved to 5.34 [8] and to 4 [6]. Recently Tran et al. [23] showed that the power assignment problem with directional antennas (described in Section 1.2) of angle $\frac{\pi}{2}$ is NP-hard, by a reduction from the Hamilton path problem on hexagonal grid graphs. A similar reduction can be employed to show that the $\frac{\pi}{2}$-MST problem is also NP-hard.

The above approximation ratios are obtained by considering the weight of the MST as the lower bound (instead of the weight of an optimal α-MST). Of these approximation ratios, the ratio 16 for $\frac{\pi}{2}$ is very interesting because for any $\alpha < \frac{\pi}{2}$ there exists a point set for which the ratio of the weight of any α-MST to the weight of any MST is $\Omega(n)$ [5]. In other words, $\alpha = \frac{\pi}{2}$ is the smallest angle for which one can obtain an α-ST of weight within some constant factor of the MST weight. However, such a factor cannot be better than 2 because for points uniformly distributed on a line the weight of α-MST could be arbitrary close to 2 times the weight of MST, for any $\alpha < \pi$ [3, 8].

1.1 Our contributions

We present an algorithm that finds a $\frac{\pi}{2}$-ST of weight at most 10 times the MST weight (Theorem 6). Thus we obtain a 10-approximation algorithm for the $\frac{\pi}{2}$-MST problem, improving upon the previous best known ratio of 16 due to Anscher and Katz [3]. Both our algorithm and that of [3] take linear time after computing an MST.

Towards obtaining the approximation ratio 10 we extend another interesting result of Aschner et al. [5] which ensures the connectivity of two sets of oriented four points that are separated by a straight line. Our extension (which is given in Theorem 3) relaxes the linear separability constraint. Most of the paper is devoted to proving this theorem.

1.2 Some related problems

There is a relationship between bounded-angle spanning trees and bounded-degree spanning trees which have received a considerable attention [7, 12, 17, 19, 20, 22]. A degree-k ST is a spanning tree in which every vertex has degree at most k. It is easily seen that any degree-k ST is an α-ST with $\alpha = 2\pi(1 - 1/k)$ because in any degree-k ST all edges that are incident to each vertex lie in some wedge of angle $2\pi(1 - 1/k)$.

The α-bottleneck spanning tree (α-BST) is a closely related problem in which the goal is to compute an α-ST whose longest edge length is minimum. This problem has been studied in the context of designing networks with bounded-range directional antennas, see for example the results of Aschner et al. [3, 5] for constructing hop-spanners for unit disk graphs, Dobrev et al. [14, 15] and Caragiannis et al. [10] for constructing bounded-degree strongly connected networks, and Carmi et al. [11] for constructing bounded-angle Hamiltonian paths. Another related problem in this context is “power assignment with directional antennas” where the objective is to assign each point $p \in P$ a wedge of angle α as well as a range r_p to obtain a connected symmetric communication network of minimum total power $\sum_{p \in P}(r_p)^\beta$ where $\beta \geqslant 1$ is the distance-power gradient [3, 5, 23].

Computing bounded-angle Hamiltonian paths and cycles on points in the plane is another related problem. For paths it is known that any set of points in the plane admits a Hamiltonian path with turning angles at most $\frac{\pi}{2}$ [11, 18] and this bound on the angle is tight [11, 16]. For cycles no tight bound on the angle is known. Dumitrescu et al. [16] proved that any even-size point set admits a Hamiltonian cycle with angles at most $\frac{2\pi}{3}$. The most famous conjecture in this context, due to Fekete and Woeginger [18], states that any even-size point set of at least 8 elements admits a Hamiltonian cycle with angles at most $\frac{\pi}{2}$.

1.3 Preliminaries for the algorithm

The following notations are adopted from [8]. Let w_p be a wedge in the plane having its apex at a point p. We denote the clockwise (right) boundary ray of w_p by $\overrightarrow{w_p}$ and its counterclockwise (left) boundary ray by $\overleftarrow{w_p}$. Let w_q be another wedge in the plane having its apex at a point q. If q lies in w_p then we say that p *sees* q (or q is *visible* from p). We say that p and q are *mutually visible*, denoted by $p \leftrightarrow q$, if p sees q and q sees p. In Figure 2 p and q are mutually visible. Let P be a set of points in the plane such that some wedge is placed at each point of P. The *induced mutual visibility graph* of P, denoted by $G(P)$, is a geometric graph with vertex set P that has a straight-line edge between two points $p, q \in P$ if and only if p and q are mutually visible. We use the term “orient” to refer to placement of wedges at points. We denote the sum of edge lengths of a geometric graph G by $w(G)$.

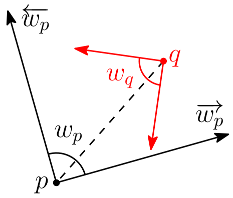


Figure 2 The points p and q are mutually visible.

We define the following notations to facilitate the description of our algorithm and its analysis. For two points p and q in the plane the *slab* $S(p, q)$ is defined as the region between two lines that are perpendicular to the segment pq at points p and q (see Figure 3(a)). We use *quadruple* to denote a set of four points in the plane. A quadruple Q is called *admissible* if it has two points p and q such that the other two points lie in $S(p, q)$ and both on the same side of pq. In this case we refer to (p, q) as an *admissible pair* of Q. Notice that a quadruple could have more than one admissible pair. For a quadruple Q with a fixed admissible pair (p, q), we define the *admissible slab* of Q, denoted by $S(Q)$, to be the same as the slab $S(p, q)$; see Figure 3(a). The following lemma (though very simple) plays an important role in our algorithm.

▶ **Lemma 1.** *Any set P of five points in the plane contains an admissible quadruple Q such that all points of P lie in $S(Q)$.*

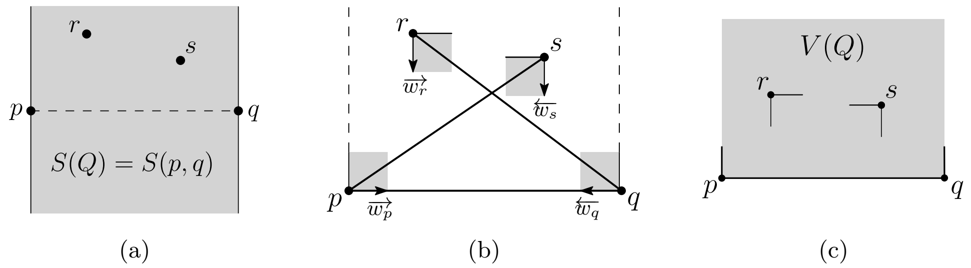


Figure 3 An admissible quadruple $Q = \{p, q, r, s\}$ with admissible pair (p, q). Illustrations of (a) the slab $S(p, q)$ which is the same as the admissible slab $S(Q)$, (b) the proof of Theorem 2, and (c) the visibility region $V(Q)$ which is the region visible to both p and q.

Proof. Let p and q be two points that define a diameter of P, i.e., two with maximum distance. Of the remaining three points of P at least two of them, say r and s, lie on the same side of $S(p, q)$. Therefore $\{p, q, r, s\}$ is an admissible quadruple which we denote by Q. Since pq is a diameter of P, all points of P lie in $S(p, q)$ and hence in $S(Q)$. ◀

Our orientation of admissible quadruples in the following theorem is similar to that of Aschner, Katz, and Morgenstern et al. [5] for arbitrary quadruples.

▶ **Theorem 2.** *Given an admissible quadruple Q, one can place at each point of Q a wedge of angle $\pi/2$ such that the wedges cover the plane and the induced mutual visibility graph of Q is connected.*

Proof. Let $Q = \{p, q, r, s\}$. After a suitable relabeling, rotation and reflection assume that (p, q) is an admissible pair of Q, the line segment pq is horizontal, p is to the left of q, the points r and s lie above pq, and r is to the left of s as in Figure 3(b). We place four wedges at points of Q as in Figure 3(b). Formally, we place a wedge w_p at p such that $\overrightarrow{w_p}$ passes through q, place w_q at q such that $\overleftarrow{w_q}$ passes through p, place w_r at r such that q lies in w_r and $\overrightarrow{w_r}$ is vertical, and place w_s at s such that p lies in w_s and $\overleftarrow{w_s}$ is vertical. These four wedges cover the entire plane (if we think of the intersection point of $\overrightarrow{w_p}$ and $\overrightarrow{w_r}$ as the origin of the coordinate system, then the four wedges cover the four quadrants). Moreover, the induced mutual visibility graph is connected because $p \leftrightarrow q$, $r \leftrightarrow q$, and $p \leftrightarrow s$. ◀

Recall the two points p and q in the proof of Theorem 2 that make Q admissible. Notice that after orientation of Theorem 2 the admissible slab of Q is uniquely defined by p and q. We define the *visibility region* of Q, denoted by $V(Q)$, as part of $S(Q)$ that is visible to both p and q; see Figure 3(c) for an illustration.

The following theorem, which will be proved in Section 3, plays a crucial role in the correctness of our algorithm. Most of the paper is devoted to proving this theorem.

▶ **Theorem 3.** *Let Q_1 and Q_2 be two admissible quadruples. Assume that wedges of angle $\pi/2$ are placed at points of each of Q_1 and Q_2 according to the placement in the proof of Theorem 2. Then at least one of the following statements holds*

(i) *The induced mutual-visibility graph of $Q_1 \cup Q_2$ is connected.*

(ii) *At any point p in $S(Q_1) \cup S(Q_2)$ one can place a wedge of angle $\pi/2$ such that p is mutually visible from a point $q_1 \in Q_1$ and from a point $q_2 \in Q_2$. In other words the induced mutual-visibility graph of $Q_1 \cup Q_2 \cup \{p\}$ is connected.*

We note that there are admissible quadruples for which statement (i) does not hold, but (ii) holds for them; see for example Figure 12. Theorem 3 extends the following result of Aschner et al. [5] which applies only to quadruples that are separated by a line.

▶ **Theorem 4** (Aschner, Katz, and Morgenstern [5], 2013). *Let Q_1 and Q_2 be two quadruples. Assume that wedges of angle $\pi/2$ are placed at points of each of Q_1 and Q_2 according to the placement in the proof of Theorem 2. If Q_1 and Q_2 are separated by a straight line, then the induced mutual-visibility graph of $Q_1 \cup Q_2$ is connected.*

2 The approximation algorithm

Let P be a set of n points in the plane. In this section we present our algorithm for computing a $\frac{\pi}{2}$-ST of P of weight at most 10 times the weight of the MST of P. In Section 2.1 we describe the general framework of the algorithm. In Section 2.2 we provide the details of the algorithm and its analysis.

2.1 A general framework

Our algorithm follows the same framework as previous algorithms [3, 6, 8] which is described below. This framework was first introduced by Aschner and Katz [3].

Start by computing an MST of P. From the MST obtain a Hamiltonian path H of weight at most 2 times the weight of MST. It is well-known that such a path can be obtained by doubling the MST edges, computing an Euler tour, and then short-cutting repeated vertices. The constant 2 is tight as Fekete et al. [17] showed that for any fixed $\varepsilon > 0$ there exist point sets for which the weight of any Hamiltonian path is at least $2 - \varepsilon$ times the weight of MST.

The next step is to partition H into $\frac{n}{k}$ groups each consisting of k consecutive vertices of H for some constant k (assuming n is divisible by k). Then orient each group independently in such a way that (I) the vertices in each group are connected, and (II) there is an edge between any pair of consecutive groups. Thus the induced mutual visibility graph on P is connected. Moreover, as the vertices of the groups are connected locally (to the vertices of the same group or a neighboring group), the mutual visibility graph contains a spanning tree whose weight is within some constant factor of the weight of H. This constant depends only on k.

The original algorithms of Aschner and Katz [3] partition H into groups of size $k = 8$ for $\alpha = \frac{\pi}{2}$ and $k = 3$ for $\alpha = \frac{2\pi}{3}$. The improved algorithms of [8] and [6] (for $\alpha = \frac{2\pi}{3}$) partition H into groups of size $k = 3$ and $k = 2$, respectively.

Our algorithm partitions H into groups of size $k = 5$ for $\alpha = \frac{\pi}{2}$. The most challenging part in our algorithm (and in previous algorithms) is to maintain property (II); the proof of this property often involves detailed case analysis. There is a main difference between our algorithm and previous algorithms [3, 6, 8]. Instead of orienting all five vertices in each group simultaneously, we first *select* four of them and orient only these selected vertices. The four selected vertices form an admissible quadruple. We refer to the non-selected vertex as a *backup*. We show that, except for one "special case", there is always a connection between two oriented admissible quadruples. For the special case we use the backup vertex to make the connection between two quadruples.

2.2 Details of our algorithm

In this section we provide details of our algorithm and its analysis. Recall that P is a set of n points in the plane, and that H is a Hamiltonian path on P such that

$$w(H) \leqslant 2w(\text{MST}).$$

Let $h_1, \ldots, h_{n-1}$ be the sequence of edges of H from one end to another. Partition the edges of H into five sets $H_1 = \{h_1, h_6, \ldots\}$, $H_2 = \{h_2, h_7, \ldots\}$, $H_3 = \{h_3, h_8, \ldots\}$, $H_4 = \{h_4, h_9, \ldots\}$, and $H_5 = \{h_5, h_{10}, \ldots\}$. Let H_k with $k \in \{1,2,3,4,5\}$ be the edge set with the largest weight. Then

$$w(H_k) \geqslant \frac{w(H)}{5} \quad \text{and} \quad w(H \setminus H_k) \leqslant \frac{4w(H)}{5}.$$

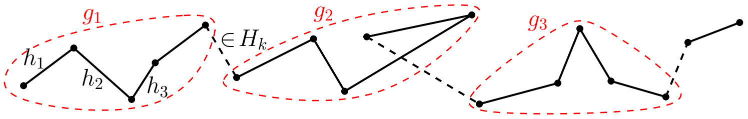


Figure 4 Illustration of the groups and sub-paths (dashed edges belong to H_k, where $k = 5$).

By removing all edges of H_k from H we obtain a sequence of sub-paths each containing five vertices (except possibly the first and last sub-paths). To simplify our description we assume for now that all sub-paths have five vertices, later in Remark 5 we will take care of the case where the first and last sub-paths have less than five vertices. We refer to the five vertices of each sub-path as a *group*. Let $g_1, g_2, \ldots, g_m$ denote the sequence of the groups that is corresponding to the sequence of sub-paths along H as in Figure 4.

From each group g_i we take an admissible quadruple Q_i (consisting of four vertices) as in the proof of Lemma 1. We denote the remaining vertex of g_i by b_i; this is a backup vertex. By Lemma 1, b_i lies in $S(Q_i)$. We orient each admissible quadruple Q_i according to the orientation in the proof of Theorem 2 which ensures the connectivity of the induced mutual visibilty graph $G(Q_i)$. Consider any two consecutive oriented quadruples Q_i and Q_{i+1}. By Theorem 3 at least one of the following statements holds:

(i) The graph $G(Q_i \cup Q_{i+1})$ is connected, i.e., there is an edge between Q_i and Q_{i+1}.

(ii) Any point p in $S(Q_i) \cup S(Q_{i+1})$ can be oriented so that $G(Q_i \cup Q_{i+1} \cup \{p\})$ is connected.

If statement (i) holds then we orient b_i towards a vertex of Q_i that sees b_i (such a vertex exists because the orientation of Theorem 2 covers the entire plane). If (i) does not hold but (ii) holds then we orient b_i in such a way that it connects Q_i and Q_{i+1}.

To this end all vertices are oriented except the backup vertex b_m of g_m. We orient b_m towards a vertex of Q_m that sees b_m. Thus, we obtain a connected induced mutual visibility graph $G(P)$.

Now we obtain a spanning tree T of $G(P)$ as follows: First we take an arbitrary spanning tree T_i from each $G(Q_i)$. Then we connect each pair T_i and T_{i+1} either by a direct edge (if (i) holds) or via a backup vertex (if (ii) holds). Lastly we connect any remaining backup vertex to its corresponding quadruple by an edge. This gives a spanning tree T that we report as the output of our algorithm. Notice that the trees T_i are not necessarily minimum spanning trees of graphs $G(Q_i)$; we will use the triangle inequality to bound the length of T.

Analysis of the approximation ratio. To bound the weight of T, we charge the edges of H for the edges of T as follows. By the triangle inequality, the weight of every edge (p, q) of T is at most the weight of the unique path in H between p and q. We charge the weight of the edges of this path for the edge (p, q). Every edge of H_k is charged only once and that is for connecting two consecutive trees T_i and T_{i+1} (either directly or via a backup vertex). Every edge of $H \setminus H_k$ (i.e., every edge of each sub-path) is charged at most six times: three times for the three edges of T_i, two times for the two edges connecting T_i to T_{i+1} and to T_{i-1}, and once for the edge connecting the backup vertex b_i to T_i. Therefore

$$\begin{aligned} w(T) &\leqslant w(H_k) + 6w(H \setminus H_k) \\ &= w(H) + 5w(H \setminus H_k) \leqslant w(H) + 5 \cdot \frac{4w(H)}{5} = 5w(H) \leqslant 10w(\mathrm{MST}). \end{aligned}$$

Running-time analysis. After computing an MST in $O(n \log n)$ time, the rest of the algorithm (computing H, finding H_k, orienting admissible quadruples and backup vertices, and obtaining T) takes $O(n)$ time.

▶ Remark 5. Here we handle the case where the first sub-path, denoted by δ, has less than five vertices (the last sub-path will be treated analogously). This case is essentially a simple version of Theorem 3 where fewer points are involved. We will use Theorem 3 to handle this case, however it could also be handled directly but with some case analysis.

We will connect the vertices of δ to g_1 (the first 5-vertex group). Let Q be g_1's admissible quadruple. Since the oriented points in Q cover the entire plane, it might be tempting to orient each point p of δ towards the point of Q that sees p. This approach may not be suitable when δ has more than one point because to maintain the ratio 10 we should not connect Q to its proceeding group (here to δ) by more than one edge. To remedy this, we use our Theorem 3.

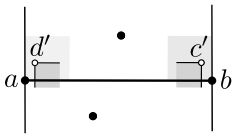


Figure 5 ab is the diameter of δ, and c', d' are fake points.

As discussed above, we may assume that δ has 2, 3, or 4 points. Let ab be a diameter of δ as in Figure 5. Thus, δ has points a, b, and at most two other "real" points. We place a "fake" point c' in $S(a, b)$ and very close to b such that both c' and b lie on the same side of any line through boundary rays of wedges in Q. In the same fashion we place a fake point d' very close to a, and on the same side of ab as c'. Let $Q' = \{a, b, c', d'\}$. Our placement of c' and d' – in $S(a, b)$ and on the same side of ab – implies that Q' is an admissible quadruple with admissible pair (a, b). We orient Q' according to Theorem 2. By Theorem 3-part (i), a point of Q' and a point of Q are mutually visible (our placement of c' and d' together with Property 1 from the next section imply that part (i) of Theorem 3 holds). If the visibility is through a real point say b, then we reflect the orientation of a with respect to ab. After reflection, a and b remain mutually visible, and their wedges cover the entire region $S(a, b)$. Then we orient every other real vertex of δ towards the one of a and b that sees it. If the visibility is through a fake point say c' then the point of Q, say q, that sees c' also sees b (this is implied by our placement of c'). In this case we reflect the orientation of b with respect to ab so that b is mutually visible with q, and a and b together see the entire region $S(a, b)$.

Then we orient every other real vertex of δ towards the one of a and b that sees it. In either case we remove fake points. Therefore the mutual visibility graph on points of δ is connected, and it has a connection to a point in Q via a or b.

The following theorem summarizes our main result.

▶ **Theorem 6.** *For any set of points in the plane and any angle $\alpha \geqslant \frac{\pi}{2}$, there is an α-spanning tree of length at most 10 times the length of the MST. Furthermore, there is an algorithm that finds such an α-spanning tree in linear time after construction of the MST.*

3 Proof of Theorem 3

In this section we prove Theorem 3 which says: *Let Q_1 and Q_2 be two admissible quadruples. Assume that wedges of angle $\pi/2$ are placed at points of each of Q_1 and Q_2 according to the placement in the proof of Theorem 2. Then at least one of the following statements holds*

(i) *The induced mutual-visibility graph of $Q_1 \cup Q_2$ is connected.*

(ii) *At any point p in $S(Q_1) \cup S(Q_2)$ we can place a wedge of angle $\pi/2$ such that p is mutually visible from a point $q_1 \in Q_1$ and from a point $q_2 \in Q_2$. In other words the induced mutual-visibility graph of $Q_1 \cup Q_2 \cup \{p\}$ is connected.*

Our proof is involved. For a better understanding we split our proof into smaller pieces based on the relative position of admissible pairs of Q_1 and Q_2. Let $Q_1 = \{a, b, c, d\}$ and $Q_2 = \{a', b', c', d'\}$. After a suitable relabeling assume that (a, b) and (a', b') are the admissible pairs of Q_1 and Q_2, respectively, that are considered in the orientation of Theorem 2. Also assume that – after the orientation of Theorem 2 – c looks towards a while d looks towards b, and similarly c' looks towards a' while d' looks towards b' as in Figures 7-13. We use this notation throughout our proof without further mentioning. Up to symmetry we have the following four cases:

A. $a'b'$ intersects ab.

B. The extension of $a'b'$ intersects the extension of ab.

C. The extension of $a'b'$ intersects ab.

D. $a'b'$ is parallel to ab.

After a suitable rotation we assume that ab is horizontal and a is to the left of b. We denote by ℓ the line through ab and by ℓ' the line through $a'b'$ as in Figure 7(a). For a point x we denote by ℓ_x the line through x that is perpendicular to ℓ, and denote by ℓ'_x the line through x that is perpendicular to ℓ. For a line l in the plane we use the terms "above" and "below" to refer to the two half planes on the two sides of l. If l is vertical then "below" refers to the left-side half plane and "above" refers to the right-side half plane. Throughout our proof, we use the following obvious observation about mutual visibility without mentioning it in all occurrences.

▶ **Observation 7.** *Assume that wedges w_p and w_q of angles $\frac{\pi}{2}$ are placed at two points p and q. If the clockwise (resp. counterclockwise) boundary ray of w_p meets the counterclockwise (resp. clockwise) boundary ray of w_q at an obtuse or a right angle then p and q are mutually visible. See Figure 6.*

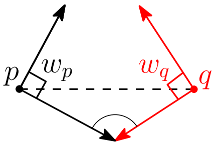
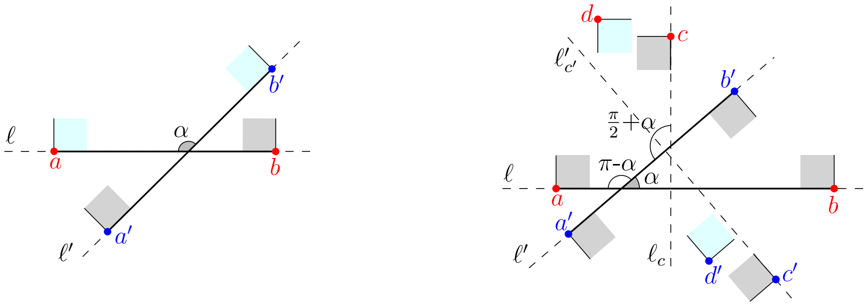


Figure 6 Illustration of Observation 7.

Some part of our proof (where Q_1 and Q_2 are separated by a line) could be implied from Theorem 4. However, for the sake of completeness we provide our own proof. We provide the proof of the first cases, A and B-1, with more formal details. To simplify our description, we will refer to the clockwise (resp. counterclockwise) boundary ray of the wedge that is placed at a point p by "the clockwise (resp. counterclockwise) ray of p".



(a) A-1

(b) A-2: c' below ℓ, c above ℓ'

Figure 7 Illustration of the proof of case A.

A. $a'b'$ intersects ab

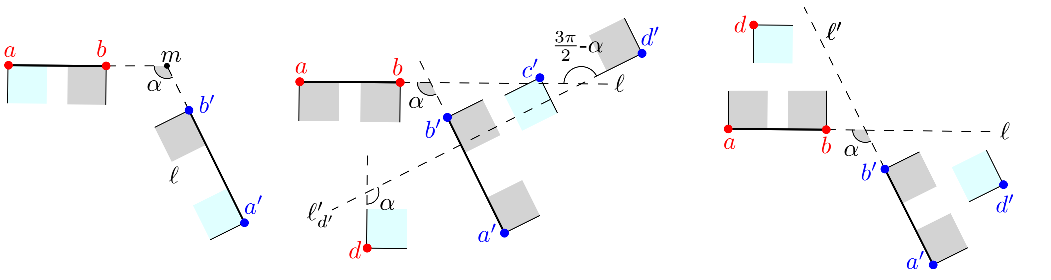
We denote by α the intersection angle of ab and $a'b'$ that lies in $V(Q_1) \cap V(Q_2)$. We say that α is *defined* by the two vertices that lie on this angle. For example in Figure 7(a) the angle α is defined by a and b'. Depending on the value of α we consider the following two cases.

1. $\alpha \geqslant \frac{\pi}{2}$. After a suitable relabeling we assume that α is defined by a and b', as in Figure 7(a). In this case the clockwise ray of a and the counterclockwise ray of b' meet at angle α, and thus a and b' are mutually visible by Observation 7.
2. $\alpha < \frac{\pi}{2}$. After a suitable relabeling we assume that α is defined by b and b', as in Figure 7(b). If c' is above ℓ then the clockwise ray of a and the counterclockwise ray of c' meet at angle $\pi - \alpha$, and thus c' and a are mutually visible by Observation 7. Similarly if c is below ℓ' then c and a' are mutually visible. Assume that c' is below ℓ and c is above ℓ' as in Figure 7(b). If d' is to the left of ℓ_c then the clockwise ray of d' and the counterclockwise ray of c meet at angle $\frac{\pi}{2} + \alpha$, and thus d' and c are mutually visible by Observation 7. Similarly if d is below $\ell'_{c'}$ then d and c' are mutually visible. Assume that d' is to the right of ℓ_c, and d is above $\ell'_{c'}$. In this setting which is depicted in Figure 7(b), d and d' lie in opposite cones formed by intersection of ℓ_c and $\ell'_{c'}$, and thus d and d' are mutually visible (observe that the clockwise ray of d and the counterclockwise ray of d' meet at angle $\pi - \alpha$).

B. The extension of $a'b'$ intersects the extension of ab

Let α be the angle at which the extensions of ab and $a'b'$ meet each other as in Figures 8 and 9. After a suitable reflection and relabeling we assume that $a'b'$ lies below ℓ, their extensions meet at a point m to the right of b, and a' is farther from m than b'. Depending on the value of α we consider two cases.

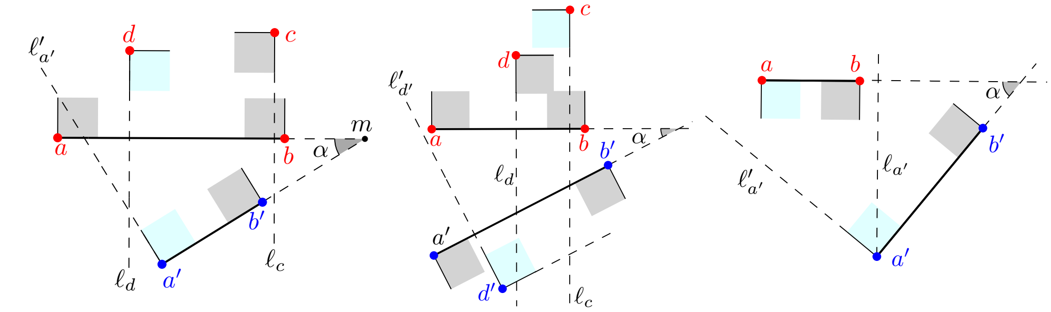
1. $\alpha \geqslant \frac{\pi}{2}$. Depending on visibility regions of Q_1 and Q_2 we consider three sub-cases (up to symmetry).
    1. $V(Q_1)$ lies below ab and $V(Q_2)$ lies below $a'b'$ as in Figure 8(a). In this case the clockwise ray of a' and the counterclockwise ray of a meet at angle α, and hence $a \leftrightarrow a'$ by Observation 7.



(a) B-1-1 (b) B-1-2: d' above ℓ, d below $\ell'_{d'}$ (c) B-1-3: d' below ℓ, d below ℓ'

Figure 8 Illustration of the proof of case B-1.

2. $V(Q_1)$ lies below ab and $V(Q_2)$ lies above $a'b'$. See Figure 8(b). If d' is below ℓ then the clockwise ray of d' and the counterclockwise ray of a meet at angle α and hence $a \leftrightarrow d'$. Assume that d' is above ℓ. If d is above $\ell'_{d'}$ then the clockwise ray of d and the counterclockwise ray of d' meet at angle $\frac{3\pi}{2} - \alpha$ and thus $d \leftrightarrow d'$. Assume that d is below $\ell'_{d'}$. In this setting which is depicted in Figure 8(b) the clockwise ray of c' and the counterclockwise ray of d meet at angle α and thus $c' \leftrightarrow d$.
3. $V(Q_1)$ lies above ab and $V(Q_2)$ lies above $a'b'$. See Figure 8(c). If d' is above ℓ then $a \leftrightarrow d'$. Similarly if d is above ℓ' then $a' \leftrightarrow d$. Assume that d' is below ℓ and d is below ℓ'. In this setting which is depicted in Figure 8(c) the clockwise ray of d' and the counterclockwise ray of d meet at angle α and thus $d \leftrightarrow d'$.



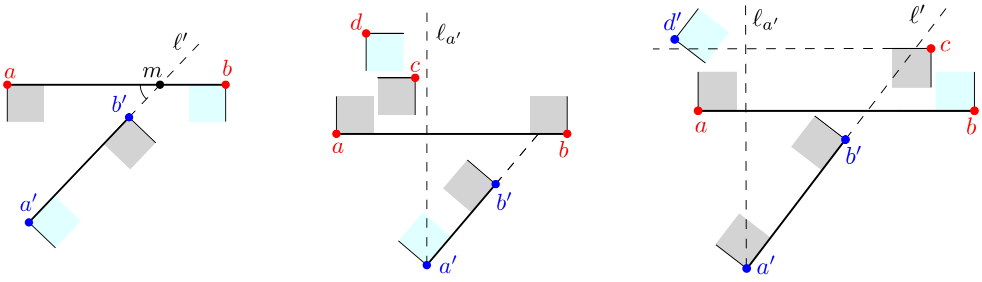
(a) B-2-1: d above $\ell'_{a'}$, a' right of ℓ_d (b) B-2-2: d' left of ℓ_d (c) B-2-3: b left of $\ell_{a'}$

Figure 9 Illustration of the proof of case B-2.

2. $\alpha < \frac{\pi}{2}$. Similar to the previous case here we also consider three sub-cases.
   1. $V(Q_1)$ lies above ab and $V(Q_2)$ lies above $a'b'$. See Figure 9(a). If d is below $\ell'_{a'}$ then d and b' are mutually visible. If a' is to the left of ℓ_d then a' and c are mutually visible. Assume that d is above $\ell'_{a'}$ and a' is to the right of ℓ_d as in Figure 9(a). In this setting d and a' are mutually visible.
   2. $V(Q_1)$ lies above ab and $V(Q_2)$ lies below $a'b'$. If d' is to the left of ℓ_d then $c \leftrightarrow d'$ as in Figure 9(b). Analogously if d is below $\ell'_{d'}$ then $c' \leftrightarrow d$. Therefore assume that d' is to right of ℓ_d and d is above $\ell'_{d'}$. In this setting $d \leftrightarrow d'$.
   3. $V(Q_1)$ lies below ab and $V(Q_2)$ lies above $a'b'$. See Figure 9(c). Consider $\ell_{a'}$, i.e., the line through a' that is perpendicular to ℓ. If b is to the right of $\ell_{a'}$ then $a' \leftrightarrow b$. Assume that b is to the left of $\ell_{a'}$ as in Figure 9(c). Now we look at $\ell'_{a'}$. If a is above this line then $a \leftrightarrow a'$, otherwise $a \leftrightarrow b'$. (Notice that when a is above $\ell'_{a'}$ then a and b' may not be mutually visible, for example when b' is very close to a'.)

C. The extension of $a'b'$ intersects ab

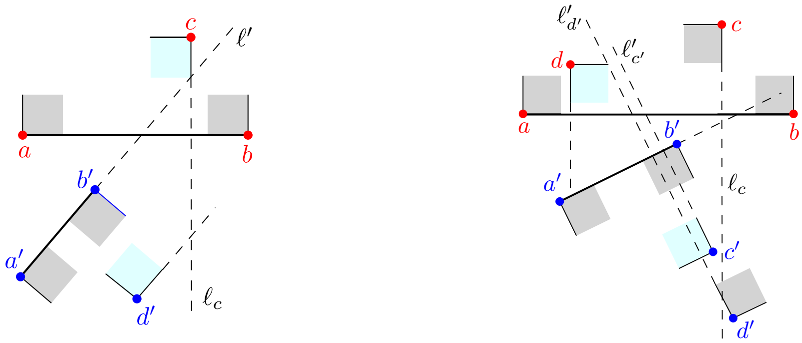
We denote by m the intersection point of ℓ' and ab. After a suitable reflection and relabeling we assume that $a'b'$ lies below ℓ, a' is farther from m than b', and angle $\angle a'ma \leqslant \frac{\pi}{2}$, as in Figure 10. Depending on visibility regions of Q_1 and Q_2 we consider four cases.



(a) C-1 (b) C-2: c left of $\ell_{a'}$ (c) C-2: c below ℓ'

Figure 10 Illustration of the proof of cases C-1 and C-2.

1. $V(Q_1)$ lies below ab and $V(Q_2)$ lies below $a'b'$ as in Figure 10(a). In this case $a' \leftrightarrow b$.
2. $V(Q_1)$ lies above ab and $V(Q_2)$ lies above $a'b'$. If c is to the left of $\ell_{a'}$ then so is d, as in Figure 10(b). In this case d sees both a' and b', and at least one of a' and b' sees d, and thus $d \leftrightarrow a'$ or $d \leftrightarrow b'$. Assume that c is to the right of $\ell_{a'}$. If c is above ℓ' then $c \leftrightarrow a'$. Thus, assume that c is below ℓ' as in Figure 10(c). Recall that d' is in slab $S(a', b')$. If d' is above the horizontal line through c then $d' \leftrightarrow b$, otherwise $d' \leftrightarrow c$.



(a) C-3: d' left of ℓ_c (b) C-3: d' right of ℓ_c, d below $\ell'_{d'}$

Figure 11 Illustration of the proof of case C-3.

3. $V(Q_1)$ lies above ab and $V(Q_2)$ lies below $a'b'$. This case is depicted in Figure 11. If c is below ℓ' then $c \leftrightarrow a'$. Assume that c is above ℓ'. If d' is to the left of ℓ_c then $c \leftrightarrow d'$ as in Figure 11(a). Assume that d' is to the right of ℓ_c (and hence to the right of ℓ_d). Now we look at d with respect to $\ell'_{d'}$. If d is above $\ell'_{d'}$ then $d \leftrightarrow d'$. If d is below $\ell'_{d'}$ then it is also below $\ell'_{c'}$ and thus $d \leftrightarrow c'$ as in Figure 11(b).
4. $V(Q_1)$ lies below ab and $V(Q_2)$ lies above $a'b'$. This case is depicted in Figure 12. If d' is below ℓ then $d' \leftrightarrow b$. Assume that d' is above ℓ. If a is below $\ell'_{b'}$ then $a \leftrightarrow b'$. Assume that a is above $\ell'_{b'}$. If c is above $\ell'_{d'}$ then $c \leftrightarrow d'$. Assume that c is below $\ell'_{d'}$ (which is also below $\ell'_{c'}$). Notice that c' lies in the slab bounded by $\ell'_{b'}$ and $\ell'_{d'}$. If c' is to the left

of ℓ_c then $c' \leftrightarrow c$. Assume that c' is to the right of ℓ_c. Notice that d lies in the vertical slab bounded by ℓ_a and ℓ_c. Let ℓ_1 be the line through c' parallel to ℓ'. If d is below ℓ_1 then $d \leftrightarrow c'$. Assume that d is above ℓ_1. This configuration is depicted in Figure 12 (the caption of this figure summarizes the constraints). This is the configuration for which statement (i) of the theorem does not hold; for all other configurations statement (i) holds. We will show that statement (ii) holds in the current setting.

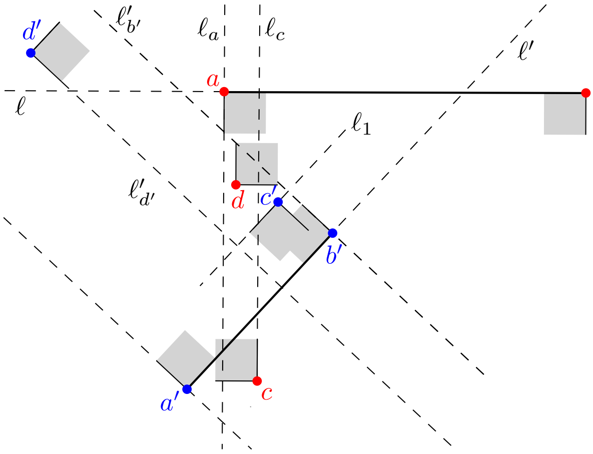


Figure 12 Illustration of case C-4: d' is above ℓ, a is above $\ell'_{b'}$, c is below $\ell'_{d'}$, c' is to the right of ℓ_c (and in the slab defined by $\ell'_{d'}$ and $\ell'_{b'}$), and d is above ℓ_1 (and in the slab defined by ℓ_a and ℓ_c). In this figure, Q_1 and Q_2 are oriented according to Theorem 2 but there is no mutual visibility between points of Q_1 and points of Q_2 (statement (i) in Theorem 3 does not hold here).

First, we extract a property of the current setting which is used in Remark 5. See Figure 12 for a better understanding of this property, and notice that in the current setting the points b, c lie on different sides of $\ell'_{b'}$, and the points a', d' lie on different sides of ℓ.

▶ **Property 1.** *If statement* (i) *in Theorem 3 does not hold then then the points b, c or the points a, d of Q_1 lie on different sides of a line through boundary rays of wedges of Q_2, and similarly the points b', c' or the points a', d' of Q_2 lie on different sides of a line through boundary rays of wedges of Q_1.*

To verify that statement (ii) holds in the current setting, let p be any point in the region $S(Q_1) \cup S(Q_2)$. We show how to place a wedge of angle $\frac{\pi}{2}$ at p so that p is mutually visible from a point in Q_1 and a point in Q_2. To simplify our description we partition $S(Q_1) \cup S(Q_2)$ into eight regions $R_1, \dots, R_8$ as in Figure 13. If $p \in R_1$ then we orient p similar to d', and thus $p \leftrightarrow b$ and $p \leftrightarrow b'$. If $p \in R_2$ then we orient p similar to a, and thus $p \leftrightarrow c$ and $p \leftrightarrow b'$. If $p \in R_3$ then we orient it similar to c' so that $p \leftrightarrow c$ and $p \leftrightarrow a'$. If $p \in R_4$ then we orient it similar to b so that $p \leftrightarrow d$ and $p \leftrightarrow a'$. If $p \in R_5$ then we orient it similar to b', and hence $p \leftrightarrow d$ and $p \leftrightarrow d'$. If $p \in R_6$ then we orient it similar to c, and thus $p \leftrightarrow a$ and $p \leftrightarrow d'$. If $p \in R_7$ then we orient it similar to a' so that $p \leftrightarrow a$ and $p \leftrightarrow c'$. Finally if $p \in R_8$ then we orient it similar to d, and hence $p \leftrightarrow b$ and $p \leftrightarrow c'$. Thus statement (ii) of the theorem holds.

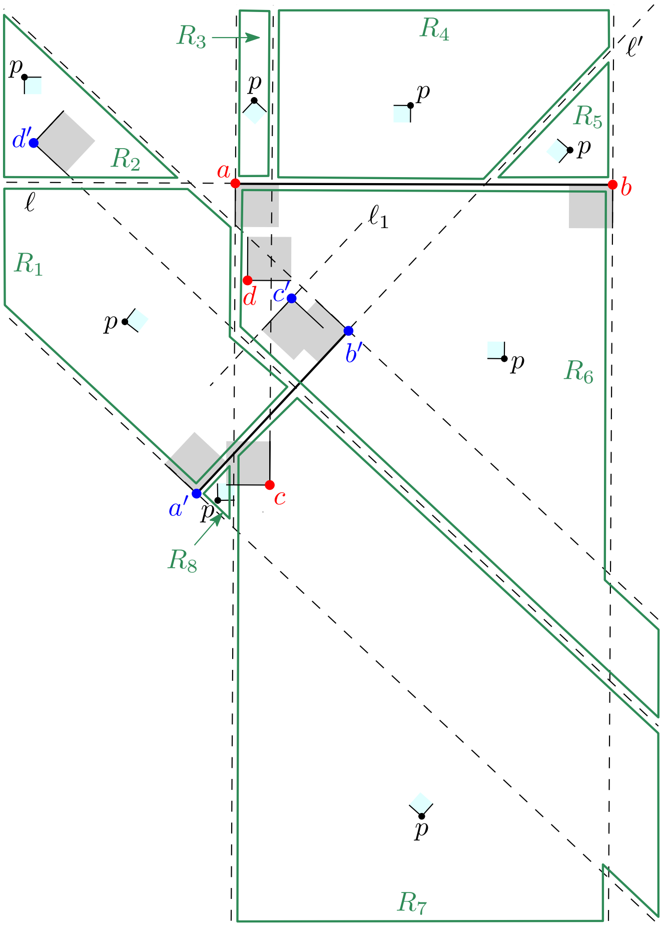


Figure 13 Partitioning $S(Q_1) \cup S(Q_2)$ into regions $R_1, \ldots, R_8$.

D. $a'b'$ is parallel to ab

Assume that ab and $a'b'$ are horizontal, and ab lies above $a'b'$. Consider any horizontal line h between ab and $a'b'$. One pair of points from Q_1 (either (a, b) or (c, d)) covers the half plane below h. Also, one pair of points from Q_2 (either (a', b') or (c', d')) covers the half plane above h. One can simply verify that there is an edge between these two pairs in the induced mutual visibility graph.

This is the end of our proof of Theorem 3.

4 Conclusions

The obvious open problem is to improve our approximation ratio 10 which we think is not the best possible ratio. The use of a Hamiltonian path is a bottleneck towards our analysis as it forces a factor of 2 in the ratio. It might be possible to get better ratios by using the original MST instead of the path. Perhaps the MST may not be the best lower bound either because one may obtain a better ratio by considering the $\frac{\pi}{2}$-MST as a lower bound.

References

1 Eyal Ackerman, Tsachik Gelander, and Rom Pinchasi. Ice-creams and wedge graphs. *Computational Geometry: Theory and Applications*, 46(3):213–218, 2013.

2 Oswin Aichholzer, Thomas Hackl, Michael Hoffmann, Clemens Huemer, Attila Pór, Francisco Santos, Bettina Speckmann, and Birgit Vogtenhuber. Maximizing maximal angles for plane straight-line graphs. *Computational Geometry: Theory and Applications*, 46(1):17–28, 2013. Also in *WADS'07*.

3 Rom Aschner and Matthew J. Katz. Bounded-angle spanning tree: Modeling networks with angular constraints. *Algorithmica*, 77(2):349–373, 2017. Also in *ICALP'14*.

4 Rom Aschner, Matthew J. Katz, and Gila Morgenstern. Do directional antennas facilitate in reducing interferences? In *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory* (*SWAT*), pages 201–212, 2012.

5 Rom Aschner, Matthew J. Katz, and Gila Morgenstern. Symmetric connectivity with directional antennas. *Computational Geometry: Theory and Applications*, 46(9):1017–1026, 2013. Also in *ALGOSENSORS'12*.

6 Stav Ashur and Matthew J. Katz. A 4-approximation of the $\frac{2\pi}{3}$-MST. In *Proceedings of the 17th Algorithms and Data Structures Symposium (WADS)*, 2021.

7 Ahmad Biniaz. Euclidean bottleneck bounded-degree spanning tree ratios. *Discrete & Computational Geometry*, https://doi.org/10.1007/s00454-021-00286-4, 2021. Also in *SODA'20*.

8 Ahmad Biniaz, Prosenjit Bose, Anna Lubiw, and Anil Maheshwari. Bounded-Angle Minimum Spanning Trees. In *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, pages 14:1–14:22, 2020.

9 Prosenjit Bose, Paz Carmi, Mirela Damian, Robin Y. Flatland, Matthew J. Katz, and Anil Maheshwari. Switching to directional antennas with constant increase in radius and hop distance. *Algorithmica*, 69(2):397–409, 2014. Also in *WADS'11*.

10 Ioannis Caragiannis, Christos Kaklamanis, Evangelos Kranakis, Danny Krizanc, and Andreas Wiese. Communication in wireless networks with directional antennas. In *Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures* (*SPAA*), pages 344–351, 2008.

11 Paz Carmi, Matthew J. Katz, Zvi Lotker, and Adi Rosén. Connectivity guarantees for wireless networks with directional antennas. *Computational Geometry: Theory and Applications*, 44(9):477–485, 2011.

12 Timothy M. Chan. Euclidean bounded-degree spanning tree ratios. *Discrete & Computational Geometry*, 32(2):177–194, 2004. Also in *SoCG'03*.

13 Mirela Damian and Robin Y. Flatland. Spanning properties of graphs induced by directional antennas. *Discrete Mathematics, Algorithms and Applications*, 5(3), 2013.

14 Stefan Dobrev, Evangelos Kranakis, Danny Krizanc, Jaroslav Opatrny, Oscar Morales Ponce, and Ladislav Stacho. Strong connectivity in sensor networks with given number of directional antennae of bounded angle. *Discrete Mathematics, Algorithms and Applications*, 4(3), 2012. Also in *COCOA'10*.

15 Stefan Dobrev, Evangelos Kranakis, Oscar Morales Ponce, and Milan Plzík. Robust sensor range for constructing strongly connected spanning digraphs in UDGs. In *Proceedings of the 7th International Computer Science Symposium in Russia* (*CSR*), pages 112–124, 2012.

16 Adrian Dumitrescu, János Pach, and Géza Tóth. Drawing Hamiltonian cycles with no large angles. *Electronic Journal of Combinatorics*, 19(2):P31, 2012. Also in *GD'94*.

17 Sándor P. Fekete, Samir Khuller, Monika Klemmstein, Balaji Raghavachari, and Neal E. Young. A network-flow technique for finding low-weight bounded-degree spanning trees. *Journal of Algorithms*, 24(2):310–324, 1997. Also in *IPCO* 1996.

18 Sándor P. Fekete and Gerhard J. Woeginger. Angle-restricted tours in the plane. *Computational Geometry: Theory and Applications*, 8:195–218, 1997.

19 Raja Jothi and Balaji Raghavachari. Degree-bounded minimum spanning trees. *Discrete Applied Mathematics*, 157(5):960–970, 2009.

20 Samir Khuller, Balaji Raghavachari, and Neal E. Young. Low-degree spanning trees of small weight. *SIAM Journal on Computing*, 25(2):355–368, 1996. Also in *STOC'94*.

21 Evangelos Kranakis, Fraser MacQuarrie, and Oscar Morales Ponce. Connectivity and stretch factor trade-offs in wireless sensor networks with directional antennae. *Theoretical Computer Science*, 590:55–72, 2015.

22 Clyde L. Monma and Subhash Suri. Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, 8:265–293, 1992. Also in *SoCG'91*.

23 Tien Tran, Min Kyung An, and Dung T. Huynh. Antenna orientation and range assignment algorithms in directional WSNs. *IEEE/ACM Transaction on Networking*, 25(6):3368–3381, 2017. Also in *INFOCOM'16*.

On Explicit Constructions of Extremely Depth Robust Graphs

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

Mike Cinkoske
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

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

Jin Young Son
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

A directed acyclic graph $G = (V, E)$ is said to be (e, d)-depth robust if for every subset $S \subseteq V$ of $|S| \leq e$ nodes the graph $G - S$ still contains a directed path of length d. If the graph is (e, d)-depth-robust for any e, d such that $e + d \leq (1 - \epsilon)|V|$ then the graph is said to be ϵ-extreme depth-robust. In the field of cryptography, (extremely) depth-robust graphs with low indegree have found numerous applications including the design of side-channel resistant Memory-Hard Functions, Proofs of Space and Replication and in the design of Computationally Relaxed Locally Correctable Codes. In these applications, it is desirable to ensure the graphs are locally navigable, i.e., there is an efficient algorithm $\mathsf{GetParents}$ running in time $\mathsf{polylog}\,|V|$ which takes as input a node $v \in V$ and returns the set of v's parents. We give the first explicit construction of locally navigable ϵ-extreme depth-robust graphs with indegree $O(\log |V|)$. Previous constructions of ϵ-extreme depth-robust graphs either had indegree $\tilde{\omega}(\log^2 |V|)$ or were not explicit.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Mathematics of computing → Graph theory; Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Combinatorics

Keywords and phrases Depth-Robust Graphs, Explicit Constructions, Data-Independent Memory Hard Functions, Proofs of Space and Replication

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.14

Related Version *Full Version*: `https://arxiv.org/abs/2110.04190`

Funding *Jeremiah Blocki*: This author was supported in part by the National Science Foundation under NSF CAREER Award CNS-2047272 and NSF Awards CCF-1910659 and CNS-1931443.
Seunghoon Lee: This author was supported in part by the Center for Science of Information at Purdue University (NSF CCF-0939370).
The opinions in this paper are those of the authors and do not necessarily reflect the position of the National Science Foundation.

1 Introduction

A depth-robust graph $G = (V, E)$ is a directed acyclic graph (DAG) which has the property that for any subset $S \subseteq V$ of at most e nodes the graph $G - S$ contains a directed path of length d, i.e., there is a directed path $P = v_0, \ldots, v_d$ such that $(v_i, v_{i+1}) \in E$ for each $i < d$ and $v_i \in V \setminus S$ for each $i \leq d$. As an example the complete DAG $K_N = (V = [N], E = \{(i, j) : 1 \leq i < j \leq n\}$ has the property that it is (e, d)-depth-robust for any integers e, d such that $e + d \leq N$. Depth-robust graphs have found many applications in cryptography including the design of data-independent Memory-Hard Functions (e.g.,[1, 3]), Proofs of

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 14; pp. 14:1–14:11
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Space [9], Proofs of Replication [15, 11] and Computationally Relaxed Locally Correctable Codes [7]. In many of these applications it is desirable to construct depth-robust graphs with low-indegree (e.g., $\mathsf{indeg}(G) = O(1)$ or $\mathsf{indeg}(G) = O(\log N)$) and we also require that the graphs are *locally navigable*, i.e., given any node $v \in V = [N]$ there is an efficient algorithm $\mathsf{GetParents}(v)$ which returns the set $\{u : (u,v) \in E\}$ containing all of v's parent nodes in time $O(\mathsf{polylog}\, N)$. It is also desirable that the graph is (e,d)-depth robust for e,d as large as possible, e.g., the cumulative pebbling cost of a graph can be lower bounded by the product ed and in the context of Memory-Hard Functions we would like to ensure that the cumulative pebbling cost is as large as possible [5, 3]. Some cryptographic constructions rely on an even stronger notion called *ϵ-extreme depth-robust graphs* $G = (V,E)$ which have the property of being (e,d)-depth-robust for any integers e,d such that $e + d \le (1-\epsilon)N$, e.g., see [15, 14].

Erdös, Graham, and Szemeredi [10] gave a randomized construction of (e,d)-depth-robust graphs with $e,d = \Omega(N)$ and maximum indegree $O(\log N)$. Alwen, Blocki, and Harsha [2] modified this construction to obtain a locally navigable construction of (e,d)-depth-robust graphs with constant indegree 2 for $e = \Omega(N/\log N)$ and $d = \Omega(N)$. For any constant $\epsilon > 0$, Schnitger [17] constructed $(e = \Omega(N), d = \Omega(N^{1-\epsilon}))$-depth-robust graphs with constant indegree – the indegree $\mathsf{indeg}(G)$ does increase as ϵ gets smaller. These results are essentially tight as *any* DAG G which is $\left(\frac{N \cdot i \cdot \mathsf{indeg}(G)}{\log N}, \frac{N}{2^i}\right)$-reducible[1] for any $i \ge 1$ [1, 18]. If $\mathsf{indeg}(G) = o(\log N)$ then the graph cannot be (e,d)-depth robust with $e,d = \Omega(N)$ and similarly if $\mathsf{indeg}(G) = \Theta(1)$ plugging in $i = O(\log\log N)$ demonstrates that G cannot be $(e = \omega(N \log\log N/\log N), d = \omega(N))$-depth-robust.

Explicit Depth-Robust Graphs

All of the above constructions are randomized and do not yield explicit constructions of depth-robust graphs. For example, the DRSample construction of [2] actually describes a randomized distribution over graphs and proves that a graph sampled from the distribution is (e,d)-depth-robust with high probability. Testing whether a graph is actually (e,d)-depth-robust is computationally intractable [8, 6] so we cannot say that a particular sampled graph is depth-robust with 100% certainty. In fact, it might be possible for a dishonest party to build a graph $G = (V,E)$ which looks like an honestly sampled depth-robust graph but actually contains a small (secret) depth-reducing set $S \subseteq V$, i.e., such that $G - S$ does not contain any long paths. Thus, in many cryptographic applications one must assume that the underlying depth-robust graphs were generated honestly.

Li [13] recently gave an explicit construction of constant-indegree depth-robust graphs, i.e., for any $\epsilon > 0$, Li constructs a family of graphs $\{G_{N,\epsilon}\}$ such that each $G_{N,\epsilon}$ has N nodes, constant indegree, and is $(\Omega(N^{1-\epsilon}), \Omega(N^{1-\epsilon}))$-depth-robust. The construction of Li [13] is also locally navigable, but the graphs are not as depth-robust as we would like. Mahmoody, Moran, and Vadhan [14] gave an explicit construction of an ϵ-extreme depth-robust graph for any constant $\epsilon > 0$ using the Zig-Zag Graph Product constructions of [16]. However, the maximum indegree is as large as $\mathsf{indeg}(G) \le \log^3 N$. Alwen, Blocki, and Pietrzak [4] gave a tighter analysis of [10] showing that the randomized construction of [10] yields ϵ-extreme depth-robust graphs with $\mathsf{indeg}(G) = O(\log N)$ although their randomized construction is not explicit nor was the graph shown to be locally navigable.

[1] If a DAG G is not (e,d)-depth-robust we say that it is (e,d)-reducible, i.e., there exists some set $S \subseteq V$ of size e such that $G - S$ contains no directed path of length d.

1.1 Our Contributions

We give explicit constructions of ϵ-extreme depth-robust graphs with maximum indegree $O(\log N)$ for any constant $\epsilon > 0$ and we also give explicit constructions of $(e = \Omega(N/\log N)$, $d = \Omega(N))$-depth-robust graphs with maximum indegree 2. Both constructions are explicit and locally navigable. In fact, our explicit constructions also satisfy a stronger property of being δ-local expanders. A δ-local expander is a directed acyclic graph G which has the following property: for any $r, v \geq 0$ and any subsets $X \subseteq A = [v, v + r - 1]$ and $Y \subseteq B = [v+r, v+2r-1]$ of at least $|X|, |Y| \geq \delta r$ nodes the graph G contains an edge (x, y) with $x \in X$ and $y \in Y$. We remark that the construction of Computationally Relaxed Locally Correctable Codes [7] relies on a family of δ-local expanders which is a strictly stronger property than depth-robustness – for any $\epsilon > 0$, there exists a constant $\delta > 0$ such that any δ-local expander automatically becomes ϵ-extreme depth-robust [4].

1.2 Our Techniques

We first provide explicit, locally navigable, constructions of δ-bipartite expander graphs with constant indegree for any constant $\delta > 0$. A bipartite graph $G = ((A, B), E)$ with $|A| = |B| = N$ is a δ-bipartite expander if for *any* $X \subseteq A$ and $Y \subseteq B$ of size $|X|, |Y| \geq \delta N$ the bipartite graph G contains at least one edge $(x, y) \in E$ with $x \in X$ and $y \in Y$. The notion of a δ-bipartite expander is related to, but distinct from, classical notions of a graph expansion, e.g., we say that G is an (N, k, d)-expander if $\mathsf{indeg}(G) \leq k$ and for every subset $X \subseteq A$ (resp. $Y \subseteq B$) we have $|\mathsf{N}(X)| \geq (1 + d - d|X|/N)|X|$ (resp. $|\mathsf{N}(Y)| \geq (1 + d - d|Y|/N)|Y|$), where $\mathsf{N}(X)$ is defined to be all of the neighbors of X, i.e., $\mathsf{N}(X) \doteq \{y \in B : \exists x \in X \text{ s.t. } (x, y) \in E\}$. (Notation: We use $\mathsf{N}(X)$ (resp. N) to denote the neighbors of nodes in X (resp. number of nodes in a graph/bipartition).) Erdös, Graham, and Szemeredi [10] argued that a random degree k_δ bipartite graph will be a δ-bipartite expander with non-zero probability where the constant k_δ depends only on δ. As a building block, we rely on an explicit, locally navigable, construction of $(n = m^2, k = 5, d = (2 - \sqrt{3})/4)$-expander graphs for any integer m due to Gabber and Galil [12]. For any constant $\delta > 0$ we show how any (N, k, d)-expander graph G with $d < 0.5$ and $k = \Theta(1)$ can be converted into a δ-bipartite expander graph G' with N nodes and maximum indegree $\mathsf{indeg}(G') = \Theta(1)$. Intuitively, the construction works by "layering" $\ell = \Theta(1)$ copies of the (N, k, d)-expander graphs and then "compressing" the layers to obtain a bipartite graph G' with maximum indegree $k' \leq k^\ell$ – paths from the bottom layer to the top layer are compressed to individual edges.

The depth-robust graph construction of Erdös et al. [10] uses δ-bipartite expanders as a building block. By swapping out the randomized (non-explicit) construction of δ-bipartite expanders with our explicit and locally navigable construction, we obtain a family of explicit and locally navigable depth-robust graphs. Furthermore, for any $\epsilon > 0$ we can apply the analysis of Alwen et al. [4] to obtain explicit constructions of ϵ-extreme depth-robust graphs by selecting the constant $\delta > 0$ accordingly. Finally, we can apply a standard indegree reduction gadget of Alwen et al. [3] to obtain an $(e = N/\log N, d = \Omega(N))$-depth-robust graph with indegree 2.

2 Preliminaries

We use $[N] = \{1, \ldots, N\}$ to denote the set of all integers between 1 and N and we typically use $V = [N]$ to denote the set of nodes in our graph. It is often convenient to assume that $N = 2^n$ is a power of 2. Given a graph $G = (V = [N], E)$ and a subset $S \subseteq [N]$ we use $G - S$ to denote the graph obtained by deleting all nodes in S and removing any incident edges. Fixing a directed graph $G = (V = [N], E)$ and a node $v \in V$, we use $\mathsf{parents}(v) = \{u : (u,v) \in E\}$ to denote the parents of node v and we let $\mathsf{indeg}(G) = \max_{v \in [N]} |\mathsf{parents}(v)|$ denote the maximum indegree of any node in G. We say a DAG G is (e,d)-reducible if there exists a subset $S \subseteq [N]$ of $|S| \leq e$ nodes such that $G - S$ contains no directed path of length d. If G is not (e,d)-reducible we say that G is (e,d)-depth-robust.

We introduce the notion of a δ-bipartite expander graph where the concept was first introduced by [10] and used as a building block to construct depth-robust graphs. Note that the specific name "δ-bipartite expander" was not used in [10]. We follow the notation of [2, 4].

▶ **Definition 1.** *A directed bipartite graph $G = ((A,B),E)$ with $|A| = |B| = N$ is called a δ-*bipartite expander *if and only if for any subset $X \subseteq A, Y \subseteq B$ of size $|X| \geq \delta N$ and $|Y| \geq \delta N$ there exists an edge between X and Y.*

▶ Remark 2. Observe that if $G = ((A,B),E)$ is a δ-bipartite expander then for any subset $X \subseteq A$ with $|X| \geq \delta N$ we must have $|\mathsf{N}(X)| > (1-\delta)N$ where $\mathsf{N}(X) = \{y \in B : \exists x \in X \text{ s.t. } (x,y) \in E\}$ denotes the neighbors of X. If this were not the case then we could take $Y = B \setminus \mathsf{N}(X)$ and we have $|Y| \geq \delta N$ and, by definition of Y, we have no edges between X and Y contradicting the assumption that G is a δ-bipartite expander.

▶ **Definition 3.** *A directed bipartite graph $G = ((A,B),E)$ with $|A| = |B| = N$ is called an (N,k,d)-*expander *if $|E| \leq kN$ and for every subset $X \subseteq A$ (resp. $Y \subseteq B$) we have $|\mathsf{N}(X)| \geq \left[1 + d\left(1 - \frac{|X|}{N}\right)\right]|X|$ (resp. $|\mathsf{N}(Y)| \geq \left[1 + d\left(1 - \frac{|Y|}{N}\right)\right]|Y|$) where $\mathsf{N}(X) = \{y \in B : \exists x \in X \text{ s.t. } (x,y) \in E\}$ (resp. $\mathsf{N}(Y) = \{x \in A : \exists y \in B \text{ s.t. } (x,y) \in E\}$).*

Gabber and Galil [12] gave explicit constructions of $(N = m^2, k = 5, d = (2-\sqrt{3})/5)$-expanders. Lemma 4 highlights the relationship between δ-bipartite expanders and the more classical notion of (N,k,d)-expanders.

▶ **Lemma 4.** *Let $0 < d < 1$ and let $\delta = \frac{(d+2)-\sqrt{d^2+4}}{2d}$. If a directed bipartite graph $G = ((A,B),E)$ with $|A| = |B| = N$ is an (N,k,d)-expander for $d < 1$ then G is a δ-bipartite expander.*

Proof. Consider an arbitrary subset $X \subseteq A$ with $|X| \geq \delta N$ and let $Y = B \setminus \mathsf{N}(X)$. We want to argue that $|Y| < \delta N$ or equivalently $|\mathsf{N}(X)| > (1-\delta)N$. Without loss of generality, we may assume that $|X| < N$ (otherwise we have $\mathsf{N}(X) = B$ since $|\mathsf{N}(X)| \geq (1 + d(1 - |X|/N))|X| = |X| = N$). Since G is an (N,k,d)-expander, we have that $|\mathsf{N}(X)| \geq \left[1 + d\left(1 - \frac{|X|}{N}\right)\right]|X| = -\frac{d}{N}|X|^2 + (d+1)|X|$. Hence, for $N > |X| \geq \delta N$, we have that

$$\begin{aligned}|\mathsf{N}(X)| &\geq -\frac{d}{N}|X|^2 + (d+1)|X| \\ &> -\frac{d}{N}(\delta N)^2 + (d+1)\delta N \\ &\geq (1-\delta)N,\end{aligned}$$

where the middle inequality follows from the observation that when $d < 1$, the function $f(x) = -\frac{d}{N}x^2 + (d+1)x$ is an increasing function over the range $0 \leq x \leq N$ and the last inequality follows from the choice of $\delta = \frac{(d+2)-\sqrt{d^2+4}}{2d}$ since $d \geq \frac{1-2\delta}{\delta-\delta^2}$. Now fixing an arbitrary subset $Y \subseteq B$ with $|Y| \geq \delta N$ and setting $X = A \setminus \mathsf{N}(Y)$, a symmetric argument shows that $|X| < \delta N$. Thus, G is a δ-bipartite expander. ◀

3 Explicit Constructions of δ-Bipartite Expanders

In this section, we give an explicit (locally navigable) construction of a δ-bipartite expander graph for any constant $\delta > 0$. As a building block, we start with an explicit construction of $(N = m^2, k = 5, d = (2 - \sqrt{3})/4)$-expander due to Gabber and Galil [12]. Applying Lemma 4 above this gives us a δ-bipartite expander with $\delta \approx 0.492$ whenever $N = m^2$. To construct depth-robust graphs we need to construct δ-bipartite expanders for much smaller values of δ and for arbitrary values of N, i.e., not just when $N = m^2$ is a perfect square. We overcome the first challenge by layering the $(N = m^2, k, d)$-expanders of [12] to obtain δ-bipartite expanders for arbitrary constants $\delta > 0$ – the indegree increases as δ approaches 0. We overcome the second issues simply by truncating the graph, i.e., if G is a $\delta/2$-bipartite expander with $2N$ nodes then we can discard up to $N/2$ sources and $N/2$ sinks and the remaining graph will still be a δ-expander.

3.1 Truncation

By layering the (N, k, d)-expanders of Gabber and Galil [12] we are able to obtain a family $\{G_{m,\delta}\}_{m=1}^{\infty}$ of δ-bipartite expanders for any constant $\delta > 0$ such that G_m has $N = m^2$ nodes on each side of the bipartition and constant indegree. However, our constructions of depth-robust graphs will require us to obtain a family $\{H_{N,\delta}\}_{N=1}^{\infty}$ of δ-bipartite expanders such that $H_{N,\delta}$ has N nodes on each side of the bipartition and constant indegree. In this section, we show how the family $\{H_{N,\delta}\}_{N=1}^{\infty}$ can be constructed by truncating graphs from the family $\{G_{m,\delta}\}_{m=1}^{\infty}$. Furthermore, if the construction of $G_{m,\delta}$ is explicit and locally navigable then so is $H_{N,\delta}$.

For each N we define $m(N) := \min_{m:m^2 \geq N}$ to be the smallest positive integer m such that $m^2 \geq N$. We first observe that for all integers $N \geq 1$ we have $m(N)^2 \geq N \geq m(N)^2/2$.

▷ Claim 5. For all $N \geq 1$ we have $m(N)^2 \geq N \geq m(N)^2/2$.

Proof. The fact that $m(N)^2 \geq N$ follows immediately from the definition of $m(N)$. For the second part it is equivalent to show that $m(N)^2/N \leq 2$ for all $N \geq 1$. The ratio $m(N)^2/N$ is maximized when $N = (m-1)^2 + 1$ for some $m \geq 1$. Thus, it suffices to show that $\frac{m^2}{(m-1)^2+1} \leq 2$ for all $m \geq 1$ or equivalently $1 + \frac{2(m-1)}{(m-1)^2+1} \leq 2$. The function $f(m) = \frac{2(m-1)}{(m-1)^2+1}$ is maximized at $m = 2$ in which case $f(2) = 1$. For all $m \geq 2$ we have $1 + \frac{2(m-1)}{(m-1)^2+1} \leq 2$ and when $m = 1$ we have $1 + \frac{2(m-1)}{(m-1)^2+1} = 1 \leq 2$ so the claim follows. ◁

Suppose that for any constant $\delta > 0$ we are given an explicit locally navigable family $\{G_{m,\delta}\}_{m=1}^{\infty}$ of δ-bipartite expanders with $G_{m,\delta} = ((A_{m,\delta} = \{X_1, \ldots, X_{m^2}\}, B_{m,\delta} = \{Y_1, \ldots, Y_{m^2}\}), E_{m,\delta})$ with edge set $E_{m,\delta} = \{(X_i, Y_j) : i \in \mathsf{GetParents}(m, \delta, j) \wedge j \leq m^2\}$ defined by an algorithm $\mathsf{GetParents}(m, \delta, j)$. We now define the algorithm $\mathsf{GetParentsTrunc}(N, \delta, j) = \mathsf{GetParents}(m(N), \delta/2, j) \cap \{1, \ldots, N\}$ and we define $H_{m,\delta} = ((A'_{N,\delta} = \{a_1, \ldots, a_N\}, B'_{N,\delta} = \{b_1, \ldots, b_N\}), E'_{N,\delta})$ with edge set $E'_{N,\delta} = \{(a_i, b_j) : i \in \mathsf{GetParentsTrunc}(N, \delta, j) \wedge j \leq N\}$. Intuitively, we start with a $\delta/2$-bipartite expander $G_{m,\delta/2}$ with $N' = m(N)^2$ nodes on each side of the partition and drop $N' - N \leq N'/2$ nodes from each side of the bipartition to obtain $H_{m,\delta}$. Clearly, if $\mathsf{GetParents}$ can be evaluated in time $O(\mathsf{polylog}\, m)$ then $\mathsf{GetParentsTrunc}$ can be evaluated in time $O(\mathsf{polylog}\, N)$. Thus, the family $\{H_{N,\delta}\}_{N=1}^{\infty}$ is explicit and locally navigable. Finally, we claim that $H_{m,\delta}$ is a δ-bipartite expander.

▶ **Lemma 6.** *Assuming that $G_{m,\delta}$ is a δ-bipartite expander for each $m \geq 1$ and $\delta > 0$, the graph $H_{m,\delta}$ is a δ-bipartite expander for each $m \geq 1$ and $\delta > 0$.*

Proof. Consider two sets $X \subseteq \{1,\ldots,N\}$ and $Y \subseteq \{1,\ldots,N\}$ and set $m = m(N)$. If $|X| \geq \delta N$ and $|Y| \geq \delta N$ then by Claim 5 we have $|X| \geq (\delta/2)m^2$ and $|Y| \geq (\delta/2)m^2$. Thus, since $G_{m,\delta/2}$ is a $\delta/2$-bipartite expander and $X, Y \subseteq \{1,\ldots,m^2\}$ there must be some pair $(i,j) \in X \times Y$ with $i \in \mathsf{GetParents}(m, \delta/2, j)$. Since $i \leq N$ we also have $i \in \mathsf{GetParentsTrunc}(N,\delta,j) = [N] \cap \mathsf{GetParents}(m,\delta/2,j)$. Thus, the edge (a_i, b_j) still exists in the truncated graph $H_{m,\delta}$. It follows that $H_{m,\delta}$ is a δ-bipartite expander. ◀

In the remainder of this section, we will focus on constructing $G_{m,\delta}$. In the next subsection, we first review the construction of $(N = m^2, k = 5, d = (2-\sqrt{3})/4)$-expanders due to Gabber and Galil [12].

3.2 Explicit (N, k, d)-Expander Graphs

Let $P_m \doteq \{0,1,\ldots,m-1\} \times \{0,1,\ldots,m-1\}$ be the set of pairs of integers (x,y) with $0 \leq x, y \leq m-1$. We can now define the family of bipartite graphs $G_m = ((A_m, B_m), E_m)$ where $A_m = \{X_{i,j} = (i,j) : (i,j) \in P_m\}$ and $B = \{Y_{i,j} = (i,j) : (i,j) \in P_m\}$. The edge set E_m is defined using the following 5 permuatations on P_m:

$$\begin{aligned}
&\sigma_0(x,y) = (x,y),\\
&\sigma_1(x,y) = (x, x+y),\\
&\sigma_2(x,y) = (x, x+y+1),\\
&\sigma_3(x,y) = (x+y, y),\\
&\sigma_4(x,y) = (x+y+1, y),
\end{aligned}$$

where the operation $+$ is modulo m. Now we can define the edge set E_m as

$$E_m = \{(X_{i',j'}, Y_{i,j}) : \exists\ 0 \leq k \leq 4 \text{ such that } \sigma_k(i',j') = (i,j)\}.$$

Gabber and Galil [12] proved that the graph G_m is a (N,k,d)-expander with $N = m^2$ nodes on each side of the biparition $(A_m \ / \ B_m)$, $k = 5$, and $d = (2-\sqrt{3})/4$.

It will be convenient to encode nodes using integers between 1 and $N = m^2$ instead of pairs in P_m. define $\texttt{PairToInt}_m(x,y) = xm + y + 1$, a bijective function mapping pairs $(x,y) \in \{0,1,\ldots,m-1\} \times \{0,1,\ldots,m-1\}$ to integers $\{1,\ldots,m^2\}$ along with the inverse mapping $\texttt{IntToPair}_m(z) = \left(\lfloor \frac{z-1}{m} \rfloor, (z-1) \mod m\right)$. We can then redefine the permutations over the set $\{1,\ldots,m^2\}$ as follows $\sigma'_j(z) = \texttt{PairToInt}_m\left(\sigma_j\left(\texttt{IntToPair}_m(z)\right)\right)$ and we can (equivalently) redefine $G_m = ((A_m, B_m), E_m)$ where $A_m = \{X_1, \ldots, X_{m^2}\}$, $B_m = \{Y_1, \ldots, Y_{m^2}\}$ and $E_m = \{(X_i, Y_j) : 1 \leq j \leq m^2 \wedge i \in \mathsf{GetParentsGG}(m,j)\}$. Here, $\mathsf{GetParentsGG}(m,j) = \{\sigma'_0(j), \sigma'_1(j), \sigma'_2(j), \sigma'_3(j), \sigma'_4(j)\}$.

3.3 Amplification via Layering

Given that we have constructed explicit δ-bipartite expanders with constant indegree for a fixed $\delta > 0$, we will construct explicit δ-bipartite expanders with constant indegree for any arbitrarily small $\delta > 0$. The construction is recursive. As our base case we define $G^0_m = G_m = ((A_m, B_m), E_m)$ where $A_m = \{X_1, \ldots, X_{m^2}\}$, $B_m = \{Y_1, \ldots, Y_{m^2}\}$ and $E_m = \{(X_i, Y_j) : 1 \leq j \leq m^2 \wedge i \in \mathsf{GetParentsGG}(m,j)\}$ as the $(N = m^2, k = 5, d = (2-\sqrt{3})/4)$-expander of Gabber and Galil [12] and we define $\mathsf{GetParentsLayered}^1(m,j) = \mathsf{GetParentsGG}(m,j)$. We can then define $G^{i+1}_m = ((A_m, B_m), E^{i+1}_m)$ where $A_m = \{X_1, \ldots, X_{m^2}\}, B_m = \{Y_1, \ldots, Y_{m^2}\}$ and $E^{i+1}_m = \{(X_i, Y_j) : 1 \leq j \leq m^2 \wedge i \in \mathsf{GetParentsLayered}^{i+1}(m,j)\}$ where

$\mathsf{GetParentsLayered}^{i+1}(m,j) = \bigcup_{j' \in \mathsf{GetParentsGG}(m,j)} \mathsf{GetParentsLayered}^{i}(m,j')$. Intuitively, we can form the graph G_m^i by stacking i copies of the graph G_m and forming a new bipartite graph by collapsing all of the intermediate layers. See Figure 1 for an illustration.

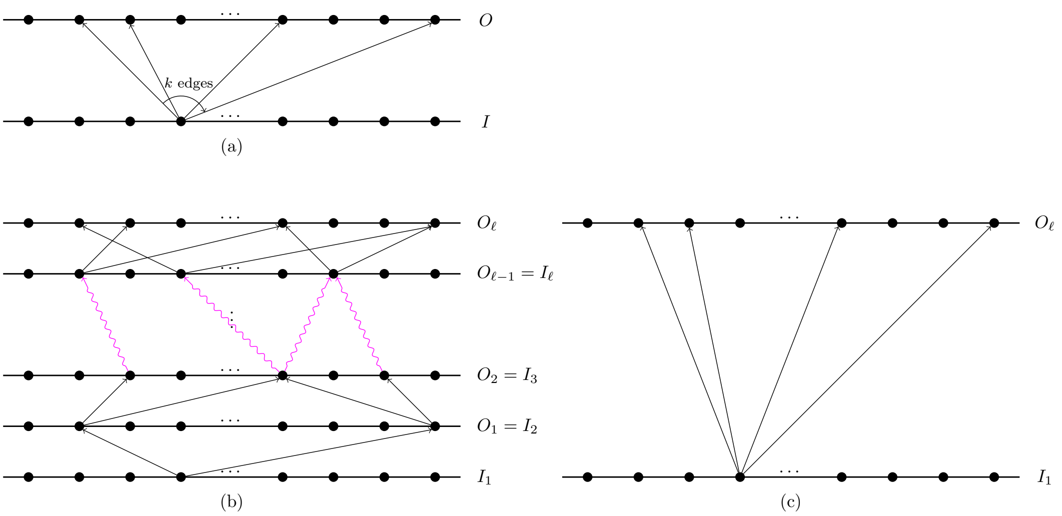


Figure 1 (a) One copy of an (N,k,d)-expander. Here, we remark that each input node has exactly k edges such that the total number of edges is kN. (b) Stack the graph ℓ times to get a graph with $(\ell+1)$ layers. The snaked edges from the third to ℓ^{th} layer indicates that there are connected paths between the nodes. (c) Generate a new bipartite graph by collapsing all of the intermediate layers. A node u on the bottom layer I_1 has an edge to a node v on the top layer O_ℓ if and only if there is a path in the original graph.

We note that $\left|\mathsf{GetParentsLayered}^{i+1}(m,j)\right| \leq k \times \left|\mathsf{GetParentsLayered}^{i}(m,j)\right| \leq k^{i+1}$. Theorem 7 tells us that amplification by layering yields a δ-bipartite expander. In particular, there is a constant L_δ such that G_m^i is a δ-bipartite expander whenever $i \geq L_\delta$. By our previous observation this graph has indegree at most k^{L_δ} which is a constant since k and L_δ are both constants.

▶ **Theorem 7.** *For any constant $\delta > 0$, there exists a constant L_δ such that for any $i \geq L_\delta$ the graph G_m^i is a δ-bipartite expander with $N = m^2$ nodes on each side of the partition.*

Proof. Fix any subset $Y^0 \subseteq [N]$ of size $|Y^0| \geq \delta N$. Let $Y^1 \doteq \bigcup_{j \in Y^0} \mathsf{GetParentsGG}(m,j)$, and recursively define $Y^{i+1} \doteq \bigcup_{j \in Y^i} \mathsf{GetParentsGG}(m,j)$. Since $Y^i = \bigcup_{j \in Y^0} \mathsf{GetParentsLayered}^i(m,j)$, it suffices to argue that $|Y^i| > (1-\delta)N$ whenever $i \geq L_\delta \doteq \left\lceil \frac{\log((1-\delta)/\delta)}{\log(1+d\delta)} \right\rceil + 1$. To see this, we note that for each $i \geq 0$, either

(1) $|Y^i|$ has already reached the target size $(1-\delta)N$, or

(2) $|Y^{i+1}| \geq \left[1 + d\left(1 - \frac{|Y^i|}{N}\right)\right]|Y^i| \geq (1+d\delta)|Y^i|$ since $\mathsf{GetParentsGG}$ defines an (N,k,d)-expander.

It follows that $|Y^{i+1}| \geq \min\{(1-\delta)N, (1+d\delta)^i \delta N\}$. Now we want to find i such that $(1+d\delta)^i \delta N = (1-\delta)N$; solving the equation we have $i = \frac{\log((1-\delta)/\delta)}{\log(1+d\delta)}$. Thus, for $i = L_\delta - 1$ we have $|Y^i| \geq (1-\delta)N$ and for $i \geq L_\delta$ we have $|Y^i| > (1-\delta)N$. Thus, for $i \geq L_\delta$ the graph G_m^i is a δ-bipartite expander, i.e., for any subsets $X, Y \subseteq [N]$ of size $|X| \geq \delta N = \delta m^2$ we must have $\left|X \cap \bigcup_{j \in Y} \mathsf{GetParentsLayered}^i(m,j)\right| > 0$ as long as $i \geq L_\delta$. ◀

3.4 Final Construction of δ-Bipartite Expanders

Based on the proof of Theorem 7, we can define $L_\delta \doteq \left\lceil \frac{\log((1-\delta)/\delta)}{\log(1+d\delta)} \right\rceil + 1$, $G_{m,\delta} \doteq G_m^{L_\delta}$, and obtain $H_{N,\delta}$ by truncating the graph $G_{m(N),\delta/2}$. The edges are defined by the procedure $\mathsf{GetParentsBE}(N,\delta,j) \doteq [N] \cap \mathsf{GetParentsLayered}^{L_{\delta/2}}(m(N),j)$ – the procedure GetParentsBE is short for "Get Parents Bipartite Expander". Formally, we have $H_{N,\delta} = ((A_N = \{a_1,\ldots,a_N\}, B_N = \{b_1,\ldots,b_N\}), E_{N,\delta})$ where $E_{N,\delta} = \{(a_i,b_j) : i \in \mathsf{GetParentsBE}(N,\delta,j)\}$.

▶ **Corollary 8.** *Fix any constant $\delta > 0$ and define $L_\delta = \left\lceil \frac{\log((1-\delta)/\delta)}{\log(1+d\delta)} \right\rceil + 1$. The graph $G_m^{L_\delta}$ is a δ-bipartite expander and the graph $H_{N,\delta}$ is a δ-bipartite expander for any integers $m, N \geq 1$.*

Proof. By Theorem 7 $G_m^{L_\delta}$ is a δ-bipartite expander. To see that $H_{N,\delta}$ is a δ-bipartite expander we simply note that $G_{m(N),\delta/2}$ is a $\delta/2$-bipartite expander and apply Lemma 6. ◀

4 Explicit Constructions of Depth Robust Graphs

We are now ready to present our explicit construction of a depth-robust graph. For any $N = 2^n$ we define the graph $G(\delta,N) = ([N], E(\delta,N))$ with edge set $E(\delta,N) = \{(u,v) : v \in [N] \wedge u \in \mathsf{GetParentsEGS}(\delta,v,N)\}$. The procedure $\mathsf{GetParentsEGS}(\delta,v,N)$ to compute the edges of $G(\delta,N)$ relies on the procedure GetParentsBE which computes the edges of our underlying bipartite expander graphs. We remark that our construction is virtually identical to the construction of [10] except that the underlying bipartite expanders are replaced with our explicit constructions from the last section.

Algorithm 1 $\mathsf{GetParentsEGS}(\delta,v,N)$.

```
procedure GetParentsEGS($\delta, v, N$)
    $P = \{v-4n, ..., v-1\}$
    for $t = 1$ to $\lceil \log_2 v \rceil$ do
        $m = \lfloor v/2^t \rfloor$
        $x = v \mod 2^t$
        $B = \mathsf{GetParentsBE}(2^t, L_{\delta/5}, x+1)$
        for $y \in B$ do
            $P = P \cup \{(m-i)2^t + y : 1 \leq i \leq \min\{m, 10\}\}$
    return $P \cap \{1, ..., N\}$
```

Note that for any constant $\delta > 0$ and any integer $n \geq 1$, the graph $G(\delta,N)$ defined by $\mathsf{GetParentsEGS}(\delta,\cdot,N)$ has $N = 2^n$ nodes and maximum indeg $\mathsf{indeg}(G(\delta,N)) = O(n) = O(\log N)$.

Erdös, Graham, and Szemeredi [10] showed that the graph $G(\delta,N)$ is a δ-local expander as long as the underlying bipartite graphs are $\delta/5$-bipartite expanders.

▶ **Theorem 9** ([10]). *For any $\delta > 0$ the graph $G(\delta,N)$ is a δ-local expander.*

Theorem 10 says that any δ-local expander is also $(e, d = N - e\frac{1+\gamma}{1-\gamma})$-depth-robust for any constant $\gamma > 2\delta$. The statement of Theorem 10 is implicit in the analysis of Alwen et al. [4]. We include the proof for completeness.

▶ **Theorem 10.** *Let $0 < \delta < 1/4$ be a constant and let $\gamma > 2\delta$. Any δ-local expander on N nodes is $(e, d = N - e\frac{1+\gamma}{1-\gamma})$-depth-robust for any $e \leq N$.*

Proof. Let G be a δ-local expander with $\delta < 1/4$ and $\gamma > 2\delta$ and let $S \subseteq [N]$ denote an arbitrary subset of size $|S| = e$. To show that $G - S$ has a path of length $d = N - e\frac{1+\gamma}{1-\gamma}$ we rely on two lemmas (Lemma 11, Lemma 12) due to Alwen et al. [4]. We first introduce the notion of a γ-good node. A node $x \in [N]$ is γ-good under a subset $S \subseteq [N]$ if for all $r > 0$ we have $|I_r(x)\backslash S| \geq \gamma |I_r(x)|$ and $|I_r^*(x)\backslash S| \geq \gamma |I_r^*(x)|$, where $I_r(x) = \{x - r - 1, ..., x\}$ and $I_r^*(x) = \{x + 1, ..., x + r\}$.

▶ **Lemma 11** ([4, 10]). *Let $G = (V = [N], E)$ be a δ-local expander and let $x < y \in [N]$ both be γ-good under $S \subseteq [N]$ then if $\delta < \min(\gamma/2, 1/4)$ then there is a directed path from node x to node y in $G - S$.*

▶ **Lemma 12** ([4]). *For any DAG $G = ([N], E)$ and any subset $S \subseteq [N]$ of nodes at least $N - |S|\frac{1+\gamma}{1-\gamma}$ of the remaining nodes in G are γ-good with respect to S.*

Applying Lemma 12 at least $d = N - e\frac{1+\gamma}{1-\gamma}$ nodes $v_1, \ldots, v_d$ are γ-good with respect to S. Without loss of generality, we can assume that $v_1 < v_2 < \ldots < v_d$. Applying Lemma 11 for each $i \leq d$, there is a directed path from v_i to v_{i+1} in $G - S$. Concatenating all of these paths we obtain one long directed path containing all of the nodes $v_1, \ldots, v_d$. Thus, $G - S$ contains a directed path of length $d = N - e\frac{1+\gamma}{1-\gamma}$. ◀

As an immediate corollary of Theorem 9 and Theorem 10 we have

▶ **Corollary 13.** *Let $0 < \delta < 1/4$ be a constant and let $\gamma > 2\delta$ then the graph $G(\delta, N)$ is $(e, d = N - e\frac{1+\gamma}{1-\gamma})$-depth-robust for any $e \leq N$.*

4.1 Explicit Extreme Depth-Robust Graphs

We also obtain explicit constructions of ϵ-extreme depth-robust graphs which have found applications in constructing Proofs of Space and Replication [15], Proofs of Sequential Work [14], and in constructions of Memory-Hard Functions [4].

▶ **Definition 14** ([4]). *For any constant $\epsilon > 0$, a DAG G with N nodes is* ϵ-extreme depth-robust *if and only if G is (e, d)-depth-robust for any $e + d \leq (1 - \epsilon)N$.*

When we set δ_ϵ appropriately the graph $G(\delta_\epsilon, N = 2^n)$ is ϵ-extremely depth robust.

▶ **Corollary 15.** *Given any constant $\epsilon > 0$ we define δ_ϵ to be the unique value such that $1 + \epsilon = \frac{1+2.1\delta_\epsilon}{1-2.1\delta_\epsilon}$ if $\epsilon \leq 1/3$ and $\delta_\epsilon = \delta_{1/3}$ for $\epsilon > 1/3$. For any integer $n \geq 1$ the graph $G(\delta_\epsilon, N = 2^n)$ is ϵ-extreme depth robust.*

Proof. Set $\gamma = 2.1\delta_\epsilon$ and observe that $\delta_{1/3} \leq 0.07 \leq 1/4$ and for $\epsilon < 1/3$ we have $\delta_\epsilon \leq \delta_{1/3} \leq 1/4$ so we can apply Corollary 13 to see that $G(\delta_\epsilon, N = 2^n)$ is $(e, d = N - e\frac{1+2.1\delta_\epsilon}{1-2.1\delta_\epsilon})$-depth robust for any $e \leq N$. Since $\frac{1+2.1\delta_\epsilon}{1-2.1\delta_\epsilon} = (1 + \epsilon)$ it follows that the graph is ϵ-extreme depth robust. ◀

4.2 Depth-Robust Graphs with Constant Indegree

In some applications it is desirable to ensure that our depth-robust graphs have constant indegree. We observe that we can apply a result of Alwen et al. [3] to transform the DAG $G(\delta, N) = (V = [N], E(\delta, N))$ with maximum indegree $\beta = \beta_{\delta,N}$ into a new DAG $H_{\delta,N} = ([N] \times [\beta], E'(\delta, N))$ with $N' = 2N\beta$ nodes and maximum indegree 2. Intuitively, the transformation reduces the indegree by replacing every node $v \in [N]$ from $G(\delta, N)$ with a path of 2β nodes $(v, 1), \ldots, (v, 2\beta)$ and distributing the incoming edges accross this path. In particular, if v has incoming edges from nodes $v_1, \ldots, v_\beta$ in $G(\delta, N)$ then for each $i \leq \beta$ we will add an edge from the node $(v_i, 2\beta)$ to the node (v, i). This ensures that each node (v, i) has at most two incoming edges. Formally, the algorithm $\mathsf{GetParentsLowIndeg}(\delta, v', N)$ takes as input a node $v' = (v, i)$ and (1) initializes $P' = \{(v, i-1)\}$ if $i > 1$, $P' = \{(v-1, 2\beta)\}$ if $i = 1$ and $v > 1$ and $P' = \{\}$ otherwise, (2) computes $P = \mathsf{GetParentsEGS}(\delta, v, N)$, (3) sets $u = P[i]$ to be the ith node in the set P, and (4) returns $P' \cup \{(u, 2\beta)\}$. It is easy to verify that the algorithm $\mathsf{GetParentsLowIndeg}$ runs in time $\mathsf{polylog}\, N$.

▶ **Corollary 16.** *Let $0 < \delta < 1/4$ be a constant and let $\gamma > 2\delta$ then the graph $H_{\delta,N}$ is $(e, d = N\beta - e\beta\frac{1+\gamma}{1-\gamma})$ depth-robust for any $e \leq N$.*

Proof. (Sketch) Alwen et al. [3] showed that applying the indegree reduction procedure above to any (e, d)-depth-robust graph with maximum indegree β yields a $(e, d\beta)$-depth-robust graph. The claim now follows directly from Theorem 9 and Theorem 10. ◀

5 Conclusion

We give the first explicit construction of ϵ-extreme depth-robust graphs $G = (V = [N], E)$ with indegree $O(\log N)$ which are locally navigable. Applying an indegree reduction gadget of Alwen et al. [3] we also obtain the first explicit and locally navigable construction of $(\Omega(N/\log N), \Omega(N))$-depth-robust graphs with constant indegree. Our current constructions are primarily of theoretical interest and we stress that we make no claims about the practicality of the constructions as the constants hidden by the asymptotic notation are large. Finding explicit and locally navigable constructions of $(c_1 N/\log N, c_2 N)$-depth-robust graphs with small indegree for reasonably large constants $c_1, c_2 > 0$ is an interesting and open research challenge. Similarly, finding explicit and locally navigable constructions of ϵ-extreme depth-robust graphs $G = (V = [N], E)$ with indegree $c_\epsilon \log N$ for smaller constants c_ϵ remains an important open challenge.

References

1 Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 241–271. Springer, Heidelberg, 2016. doi:10.1007/978-3-662-53008-5_9.

2 Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1001–1017. ACM Press, October / November 2017. doi:10.1145/3133956.3134031.

3 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56617-7_1.

4 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 99–130. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_4.

5 Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, 2015. doi:10.1145/2746539.2746622.

6 Mohammad Hassan Ameri, Jeremiah Blocki, and Samson Zhou. Computationally data-independent memory hard functions. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 36:1–36:28. LIPIcs, 2020. doi:10.4230/LIPIcs.ITCS.2020.36.

7 Jeremiah Blocki, Venkata Gandikota, Elena Grigorescu, and Samson Zhou. Relaxed locally correctable codes in computationally bounded channels. *IEEE Transactions on Information Theory*, 67(7):4338–4360, 2021. doi:10.1109/TIT.2021.3076396.

8 Jeremiah Blocki and Samson Zhou. On the computational complexity of minimal cumulative cost graph pebbling. In Sarah Meiklejohn and Kazue Sako, editors, *FC 2018*, volume 10957 of *LNCS*, pages 329–346. Springer, Heidelberg, February / March 2018. doi:10.1007/978-3-662-58387-6_18.

9 Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, 2015. doi:10.1007/978-3-662-48000-7_29.

10 P. Erdös, R.L. Graham, and E. Szemerédi. On sparse graphs with dense long paths. *Computers & Mathematics with Applications*, 1(3):365–369, 1975. doi:10.1016/0898-1221(75)90037-1.

11 Ben Fisch. Tight proofs of space and replication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 324–348. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-17656-3_12.

12 Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, 1981.

13 Aoxuan Li. On explicit depth robust graphs. *UCLA*, ProQuest ID: Li_ucla_0031N_-17780. Merritt ID: ark:/13030/m5130rq7, 2019. URL: https://escholarship.org/uc/item/4fx1m6dh.

14 Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, 2013. doi:10.1145/2422436.2422479.

15 Krzysztof Pietrzak. Proofs of catalytic space. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 59:1–59:25. LIPIcs, 2019. doi:10.4230/LIPIcs.ITCS.2019.59.

16 Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *41st FOCS*, pages 3–13. IEEE Computer Society Press, 2000. doi:10.1109/SFCS.2000.892006.

17 Georg Schnitger. On depth-reduction and grates. In *24th FOCS*, pages 323–328. IEEE Computer Society Press, 1983. doi:10.1109/SFCS.1983.38.

18 Leslie Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.

Reconfiguration of Spanning Trees with Degree Constraint or Diameter Constraint

Nicolas Bousquet
CNRS, LIRIS, Université de Lyon, France

Takehiro Ito
Graduate School of Information Sciences,
Tohoku University, Japan

Yusuke Kobayashi
Research Institute for Mathematical Sciences,
Kyoto University, Japan

Haruka Mizuta
Graduate School of Information Sciences,
Tohoku University, Japan

Paul Ouvrard
Université de Bordeaux, France

Akira Suzuki
Graduate School of Information Sciences,
Tohoku University, Japan

Kunihiro Wasa
Toyohashi University of Technology, Japan

Abstract

We investigate the complexity of finding a transformation from a given spanning tree in a graph to another given spanning tree in the same graph via a sequence of edge flips. The exchange property of the matroid bases immediately yields that such a transformation always exists if we have no constraints on spanning trees. In this paper, we wish to find a transformation which passes through only spanning trees satisfying some constraint. Our focus is bounding either the maximum degree or the diameter of spanning trees, and we give the following results. The problem with a lower bound on maximum degree is solvable in polynomial time, while the problem with an upper bound on maximum degree is PSPACE-complete. The problem with a lower bound on diameter is NP-hard, while the problem with an upper bound on diameter is solvable in polynomial time.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases combinatorial reconfiguration, spanning trees, PSPACE, polynomial-time algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.15

Related Version *Full Version*: `https://arxiv.org/abs/2201.04354`

Funding This work is partially supported by JSPS and MEAE-MESRI under the Japan-France Integrated Action Program (SAKURA).
Nicolas Bousquet: This work was supported by ANR project GrR (ANR-18-CE40-0032).
Takehiro Ito: Partially supported by JSPS KAKENHI Grant Numbers JP18H04091, JP19K11814 and JP20H05793, Japan.
Yusuke Kobayashi: Partially supported by JSPS KAKENHI Grant Numbers 18H05291, JP20K11692, and 20H05795, Japan.
Paul Ouvrard: This work was supported by ANR project GrR (ANR-18-CE40-0032).
Akira Suzuki: Partially supported by JSPS KAKENHI Grant Numbers JP18H04091, JP20K11666 and JP20H05794, Japan.
Kunihiro Wasa: Partially supported by JST CREST Grant Numbers JPMJCR18K3 and JPMJCR1401, and JSPS KAKENHI Grant Numbers 19K20350 and JP20H05793, Japan.

Acknowledgements The authors thank anonymous reviewers for their valuable comments.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 15; pp. 15:1–15:21
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

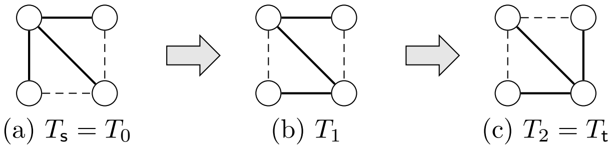


Figure 1 A reconfiguration sequence from T_{s} to T_{t} (with no constraint on spanning trees). There is no reconfiguration sequence from T_{s} to T_{t} if we restrict spanning trees either with maximum degree at least three or with diameter at most two.

1 Introduction

Given an instance of some combinatorial search problem and two of its feasible solutions, a *reconfiguration problem* asks whether one solution can be transformed into the other in a step-by-step fashion, such that each intermediate solution is also feasible. Reconfiguration problems capture dynamic situations, where some solution is in place and we would like to move to a desired alternative solution without becoming infeasible. A systematic study of the complexity of reconfiguration problems was initiated in [14]. Recently the topic has gained a lot of attention in the context of CSP and graph problems, such as the independent set problem, the matching problem, and the dominating set problem. For an overview of recent results on reconfiguration problems, the reader is referred to the surveys of van den Heuvel [11] and Nishimura [17].

In this paper, our reference problem is the spanning tree problem. Let $G = (V, E)$ be a connected graph on n vertices. A *spanning tree* of G is a subgraph of G which is a tree (connected acyclic subgraph) and includes all the vertices in G. Spanning trees naturally arise in various situations such as routing or discrete geometry. In order to define a valid step-by-step transformation, an adjacency relation on the set of feasible solutions is needed. Let T_1 and T_2 be two spanning trees of G. We say that T_1 and T_2 are *adjacent* by an *edge flip* if there exist $e_1 \in E(T_1)$ and $e_2 \in E(T_2)$ such that $E(T_2) = (E(T_1) \setminus \{e_1\}) \cup \{e_2\}$. For two spanning trees T_{s} and T_{t} of G, a *reconfiguration sequence* (or simply a *transformation*) from T_{s} to T_{t} is a sequence of spanning trees $\langle T_0 := T_{\mathsf{s}}, T_1, \ldots, T_\ell := T_{\mathsf{t}} \rangle$ such that two consecutive spanning trees are adjacent. Ito et al. [14] remarked that any spanning tree can be transformed into any other via a sequence of edge flips, which easily follows from the exchange property of the matroid bases.

In practice, we often need that spanning trees satisfy some additional desirable properties. Even if finding a spanning tree can be done in polynomial time, the problem becomes often NP-complete when additional constraints are added. In this paper, we consider spanning tree reconfiguration with additional constraints. More formally, we study the following questions: 1) does a transformation always exist when we add some constraints on the spanning trees all along the transformation? 2) If not, is it possible to decide efficiently if such a transformation exists? This question was already studied for spanning trees with restrictions on the number of leaves [2] or vertex modification between Steiner trees [16] for instance. If the answer to the first question is positive, it means that we can sample uniformly at random constrained spanning trees via a simple Monte Carlo Markov Chain. When the answer is negative, we might still want to find a transformation if possible between a fixed pair of solutions, for instance for updating a routing protocol in a network step by step without breaking the network and not over-requesting nodes during the transformation.

In this paper, we study Reconfiguration of Spanning Trees (RST) with degree constraints or with diameter constraints (See Figure 1.) We first describe the problem with degree constraints.

RST WITH SMALL (RESP. LARGE) MAXIMUM DEGREE
Input: A graph G, a positive integer d, and two spanning trees T_s and T_t in G with maximum degree at most (resp. at least) d.
Question: Is there a reconfiguration sequence from T_s to T_t such that any spanning tree in the sequence is of maximum degree at most (resp. at least) d?

Bounding the maximum degree of spanning trees has applications for routing problems when we send data (i.e., a flow) along a spanning tree in a communication network. In this setting, the degree of a node is a measure of its load, and hence it is natural to bound the maximum degree in the spanning tree. In a complex dynamic networks, we want to reconfigure spanning trees on the fly to keep this property on the dynamic setting, which motivates us to study the reconfiguration problem.

The problem of finding a spanning tree with degree bounds is studied also from the theoretical point of view. Notice that spanning trees with bounds on the maximum degree include Hamiltonian paths that are spanning trees of maximum degree two. This implies that finding a spanning tree with maximum degree at most d is NP-hard. For restricted graph classes, this search problem is investigated in [5]. It is shown in [6] that if we relax the degree bound by one, then the search problem can be solved in polynomial time. Its optimization variants are also studied in [7, 18].

We also study the problem with diameter constraints, which is formally stated as follows.

RST WITH SMALL (RESP. LARGE) DIAMETER
Input: A graph G, a positive integer d, and two spanning trees T_s and T_t in G with diameter at most (resp. at least) d.
Question: Is there a reconfiguration sequence from T_s to T_t such that any spanning tree in the sequence is of diameter at most (resp. at least) d?

Spanning trees with largest possible diameter are Hamiltonian paths which receive a considerable attention. Spanning trees with upper bound on the diameter are for instance desirable in high-speed networks like optical networks since they minimize the worst-case propagation delay to all the nodes of the graphs, see e.g. [13]. We can find a spanning tree with minimum diameter in polynomial time [9], and some related problems have been studied in the literature [8, 19]

The problem of updating minimum spanning trees to maintain a valid spanning tree in dynamic networks is an important problem that received a considerable attention in the last decades, see for instance [1, 12]. In this situation, the graph is dynamic and is dynamically updated at each time step. The solution at time t, which might not be a solution anymore at time $t+1$ (e.g. if edges of the spanning has been deleted from the graph), has to be modified with as few modifications as possible into a valid solution as good as possible. Spanning tree reconfiguration lies between the static situation (since the graph is fixed) and the dynamic situation (since the solution has to be modified).

Our Results

The contribution of this paper is to study the computational complexity of RST WITH SMALL (or LARGE) MAXIMUM DEGREE and RST WITH SMALL (or LARGE) DIAMETER.

▶ **Theorem 1.** RST WITH LARGE MAXIMUM DEGREE *can be decided in polynomial time.*

Our proof for Theorem 1 is in two steps. First we show that if there exists a vertex that has degree at least d in both T_{s} and T_{t}, then there is a reconfiguration sequence between them. Then, for two vertices u and v, we prove that we can decide in polynomial time if there exists a pair of adjacent spanning trees T and T' such that u has degree at least d in T and v has degree at least d in T'. These results together will imply Theorem 1.

While the existence of a spanning tree with maximum degree at least d can be decided in polynomial time, it is NP-complete to find a spanning tree of maximum degree at most 2 (that is a Hamiltonian path). A similar behavior holds for RST with degree constraints.

▶ **Theorem 2.** *For every $d \geq 3$,* RST WITH SMALL MAXIMUM DEGREE *is PSPACE-complete.*

The proof for Theorem 2 consists of a reduction from NCL (Nondeterministic Constraint Logic), known to be PSPACE-complete [10]. This result is tight in the following sense: if at least one of T_{s} and T_{t} has maximum degree at most $d-1$, then the problem becomes polynomial-time solvable (shown in Theorem 15). It is worth noting that this behavior is similar to the result for the search problem shown in [6]; while finding a spanning tree with maximum degree at most d is NP-hard, if we relax the degree bound by one, then the problem can be solved in polynomial time.

In the second part of the paper, we study RST WITH SMALL or LARGE DIAMETER.

▶ **Theorem 3.** RST WITH LARGE DIAMETER *is NP-hard even restricted to planar graphs.*

The proof for Theorem 3 consists of a reduction from the HAMILTONIAN PATH problem, which is not a reconfiguration problem but the original search problem. We note that since the length of a reconfiguration sequence is not necessarily bounded by a polynomial in the input size, it is unclear whether RST WITH LARGE DIAMETER belongs to the class NP. In a similar way to RST WITH SMALL MAXIMUM DEGREE, we conjecture that RST WITH LARGE DIAMETER is PSPACE-complete.

Finally, the main technical result of the paper is the following positive result.

▶ **Theorem 4.** RST WITH SMALL DIAMETER *is polynomial-time solvable.*

The proof for Theorem 4 follows a similar scheme to Theorem 1. First we show that all the spanning trees with the same "center" can be transformed into any other. Therefore, it suffices to consider the transformation of the centers. However, for two vertices u and v, it is hard to determine whether there exists a pair of adjacent spanning trees T and T' such that u and v are centers of T and T', respectively. Indeed, we do not know whether it can be done in polynomial time. The core of the proof is to focus on only "good" pairs of centers for which the existence of a desired pair of spanning trees can be tested in polynomial time (see Theorem 23). A key ingredient of our proof consists in proving that if there is a reconfiguration sequence between the spanning trees, then there exists a sequence of centers from the initial center to the final center in which any consecutive centers form a good pair (see Theorem 24).

Organization

The rest of this paper is organized as follows. We first give some preliminaries in Section 2. Next, Sections 3 and 4 are devoted to RST WITH LARGE MAXIMUM DEGREE (Theorem 1) and RST WITH SMALL MAXIMUM DEGREE (Theorem 2), respectively. Then, Sections 5 and 6 are devoted to RST WITH LARGE DIAMETER (Theorem 3) and RST WITH SMALL DIAMETER (Theorem 4), respectively. Finally, we conclude this paper by giving some remarks in Section 7. Due to the space limitation, the proofs of the statements marked with ($\star$) have been deferred to the appendix, and marked with ($\star\star$) have been deferred to the full version [3].

2 Preliminaries

Throughout this paper, we consider graphs that are simple and loopless. Let $G = (V, E)$ be a graph. For a vertex $v \in V$, we denote by $d_G(v)$ the *degree* of v in G, by $N_G(v)$ the (open) *neighborhood* of v in G, and by $\delta_G(v)$ the set of edges incident to v in G. Since G is simple, $d_G(v) = |N_G(v)| = |\delta_G(v)|$. For a tree T, a vertex v is a *leaf* if its degree is one, and is an *internal node* otherwise. A *branching node* is a vertex of degree at least three.

For a subgraph H of G and an $F \subseteq E$, we denote by $H - F$ the graph $(V(H), E(H) \setminus F)$ and by $H + F$ the graph $(V(H), E(H) \cup F)$. To avoid cumbersome notation, if $e \in E$, $H - \{e\}$ and $H + \{e\}$ will be denoted by $H - e$ and $H + e$, respectively.

For $u, v \in V$, the *distance* $\bar{\ell}_G(u, v)$ between u and v is defined as the minimum number of edges in a shortest u-v path. For $v \in V$, the *eccentricity* $\epsilon_G(v)$ of v in G is the maximum distance between v and any vertex in G, that is, $\epsilon_G(v) := \max\left\{\bar{\ell}_G(v, u) \mid u \in V\right\}$. The *diameter* $\mathsf{diam}(G)$ of G is the maximum eccentricity among V. That is, $\mathsf{diam}(G) := \max\{\epsilon_G(v) \mid v \in V\} = \max\left\{\bar{\ell}_G(u, v) \mid u, v \in V\right\}$.

For two spanning trees T and T', we denote $T \leftrightarrow T'$ if $|E(T) \setminus E(T')| = |E(T') \setminus E(T)| \leq 1$, that is, either $T = T'$ or T and T' are adjacent. We say that T_{s} is *reconfigurable* to T_{t} if there exists a reconfiguration sequence from T_{s} to T_{t} such that any spanning tree in the sequence satisfies a given degree/diameter constraint. When we have no degree/diameter constraints, since spanning trees form a base family of a matroid, the exchange property of the matroid bases ensures that there always exists a reconfiguration sequence between any pair of spanning trees.

▶ **Lemma 5** (see Proposition 1 in [14]). *Let G be a graph and T and T' be two spanning trees of G. There exists a reconfiguration sequence $\langle T = T_0, T_1, \ldots, T_\ell = T'\rangle$ between T and T' such that for all $i \in \{0, 1, \ldots, \ell\}$, the spanning tree T_i contains all the edges in $E(T) \cap E(T')$.*

3 Large Maximum Degree (Proof of Theorem 1)

In this section, we prove Theorem 1, which we restate here.

▶ **Theorem 1.** RST WITH LARGE MAXIMUM DEGREE *can be decided in polynomial time.*

Let $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ be an instance of RST WITH LARGE MAXIMUM DEGREE. For a spanning tree T in G, let $\mathsf{large}(T) \subseteq V$ be the set of all the vertices of degree at least d in T, that is, $\mathsf{large}(T) := \{v \in V \mid d_T(v) \geq d\}$. Note that T has maximum degree at least d if and only if $\mathsf{large}(T) \neq \emptyset$. The following lemma is easy but is essential to prove Theorem 1.

▶ **Lemma 6** ($\star$). *Let T_1 and T_2 be spanning trees in G with maximum degree at least d. If there exists a vertex $u \in \mathsf{large}(T_1) \cap \mathsf{large}(T_2)$, then T_1 is reconfigurable to T_2.*

Our algorithm is based on testing the reachability in an auxiliary graph $\mathcal{G}$, which is defined as follows. The vertex set of $\mathcal{G}$ is defined as V, where each vertex v in $V(\mathcal{G})$ corresponds to the set of spanning trees T with $v \in \mathsf{large}(T)$. For any pair u, v of distinct vertices in $V(\mathcal{G})$, there is an edge $uv \in E(\mathcal{G})$ if and only if there exist spanning trees T and T' such that $u \in \mathsf{large}(T)$, $v \in \mathsf{large}(T')$, and $T \leftrightarrow T'$ (possibly $T = T'$). Then, by definition of the auxiliary graph and Lemma 6, we have the following lemma.

Algorithm 1 Algorithm for RST with Large Maximum Degree.

```
Input: A graph $G$ and two spanning trees $T_s$ and $T_t$ in $G$ with max. degree $\geq d$.
Output: Is $T_s$ reconfigurable to $T_t$?
Compute $\mathsf{large}(T_s)$ and $\mathsf{large}(T_t)$, and construct $\mathcal{G}$;
if there is a path between $\mathsf{large}(T_s)$ and $\mathsf{large}(T_t)$ in $\mathcal{G}$ then return YES;
else return NO;
```

▶ **Lemma 7** (⋆). *Let T_s and T_t be spanning trees with maximum degree at least d. Then, T_s is reconfigurable to T_t if and only if $\mathcal{G}$ contains a path from $\mathsf{large}(T_s)$ to $\mathsf{large}(T_t)$.*

By this lemma, we can solve RST with Large Maximum Degree by detecting a path from $\mathsf{large}(T_s)$ to $\mathsf{large}(T_t)$ in $\mathcal{G}$ (see Algorithm 1 for a pseudocode of our algorithm). Our remaining task is to construct the auxiliary graph $\mathcal{G}$ in polynomial time which is possible by the following lemma.

▶ **Lemma 8** (⋆). *For two distinct vertices $u, v \in V$, there exists an edge $uv \in E(\mathcal{G})$ if and only if $|N_G(u)| \geq d$, $|N_G(v)| \geq d$, and*

$$|N_G(u) \cup N_G(v)| \geq \begin{cases} 2d-1 & \text{if } uv \in E(G), \\ 2d-2 & \text{otherwise.} \end{cases} \tag{1}$$

Since we can easily check the inequality (1) for each pair of vertices u and v, Lemma 8 ensures that the auxiliary graph $\mathcal{G}$ can be constructed in polynomial time. Therefore, Algorithm 1 correctly decides RST with Large Maximum Degree in polynomial time, which completes the proof of Theorem 1. Note that all the proofs are constructive, and hence we can find a desired reconfiguration sequence from T_s to T_t in polynomial time if it exists.

4 Small Maximum Degree

In this section, we consider RST with Small Maximum Degree. We first show the PSPACE-completeness in Section 4.1. In contrast, we show in Section 4.2 that if at least one of T_s and T_t has maximum degree at most $d-1$, then an instance (G, d, T_s, T_t) of RST with Small Maximum Degree is a YES-instance.

4.1 PSPACE-Completeness (Proof of Theorem 2)

In this subsection, we prove Theorem 2, i.e., we show that RST with Small Maximum Degree is PSPACE-complete. The problem is indeed in PSPACE. We prove the PSPACE-hardness by giving a polynomial reduction from *Reconfiguration of Nondeterministic Constraint Logic on* and/or *graphs*, which we call NCL Reconfiguration for short.

Suppose that we are given a cubic graph with edge-weights such that each vertex is either incident to three weight-2 edges ("or vertex") or one weight-2 edge and two weight-1 edges ("and vertex"), which we call an and/or *graph*. An *NCL configuration* is an orientation of the edges in the graph such that the total weights of incoming arcs at each vertex is at least two. Two NCL configurations are *adjacent* if they differ in a single edge direction. In NCL Reconfiguration, we are given an and/or graph and its two NCL configurations, and the objective is to determine whether there exists a sequence of adjacent NCL configurations that transforms one into the other. It is shown in [10] that NCL Reconfiguration is PSPACE-complete. In what follows, we give a polynomial reduction from NCL Reconfiguration to RST with Small Maximum Degree.

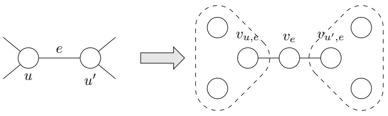


Figure 2 The gadget for an edge.

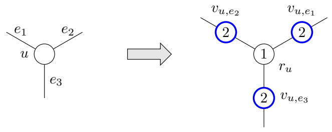


Figure 3 The gadget for an OR vertex.

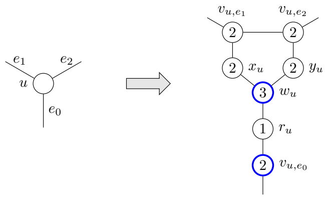


Figure 4 The gadget for an AND vertex.

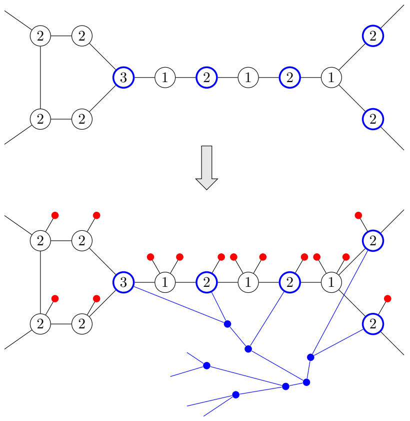


Figure 5 Construction of G from G'.

Construction of the graph. Suppose that we are given an instance of NCL RECONFIGURATION, that is, an AND/OR graph $H = (V(H), E(H))$ and two configurations σ_s and σ_t of H. Fix $d \geq 3$. We first construct a graph $G' = (V', E')$, a vertex subset $L \subseteq V'$, and an integer $b(v) \in \{1, 2, 3\}$ for each $v \in V'$, and then construct a graph $G = (V, E)$ by using G', L, and b. We consider an instance $(G, d, T_\mathsf{s}, T_\mathsf{t})$ of RST WITH SMALL MAXIMUM DEGREE, where T_s and T_t will be defined later. The construction of G', L, and b is described as follows.

- We initialize $G' = (V', E')$ and L as the empty graph and the empty set, respectively.
- For a vertex $u \in V(H)$ and an edge $e \in \delta_H(u)$, we introduce a vertex $v_{u,e}$ in V'. Let $b(v_{u,e}) = 2$.
- For an edge $e \in E(H)$ connecting u and u', we introduce a vertex v_e in V' and two edges $v_e v_{u,e}$ and $v_e v_{u',e}$ in E' (Figure 2). Let $b(v_e) = 1$.
- For an OR vertex $u \in V(H)$ with $\delta_H(u) = \{e_1, e_2, e_3\}$, we introduce a vertex r_u in V' and an edge $r_u v_{u,e_i}$ in E' for $i \in \{1, 2, 3\}$ (Figure 3). Let $b(r_u) = 1$. Add v_{u,e_i} to L for $i \in \{1, 2, 3\}$.
- For an AND vertex $u \in V(H)$ with $\delta_H(u) = \{e_0, e_1, e_2\}$, where e_0 is a weight-2 edge and e_1 and e_2 are weight-1 edges, we introduce four vertices r_u, w_u, x_u, and y_u in V', and seven edges $v_{u,e_0} r_u, r_u w_u, w_u x_u, w_u y_u, x_u v_{u,e_1}, y_u v_{u,e_2}$, and $v_{u,e_1} v_{u,e_2}$ in E' (Figure 4). We denote by E'_u the set of these seven edges. Let $b(r_u) = 1$, $b(w_u) = 3$, and $b(x_u) = b(y_u) = 2$. Add v_{u,e_0} and w_u to L.

We next construct $G = (V, E)$ by adding new vertices and edges to $G' = (V', E')$ as follows (see Figure 5 for an illustration).

- We construct a tree $T^* = (V(T^*), E(T^*))$ of maximum degree at most three such that $V(T^*) \cap V' = L$, $E(T^*) \cap E' = \emptyset$, and L is the set of all the leaves of T^*. Then, we attach T^* to G'. We denote the obtained graph by $G' + T^*$.
- For each vertex $v \in V'$, we add $d - b(v)$ new vertices $\bar{v}_1, \ldots, \bar{v}_{d-b(v)}$ and new edges $v\bar{v}_1, \ldots, v\bar{v}_{d-b(v)}$.

Correspondence between solutions. In order to see the correspondence between NCL configurations in H and spanning trees in G with maximum degree at most d, we begin with the following easy lemma.

▶ **Lemma 9.** *Any spanning tree T in G with maximum degree at most d satisfies the following properties: (a) $v\bar{v}_i \in E(T)$ for $v \in V'$ and for $i \in \{1, 2, \dots, d - b(v)\}$; (b) $|\delta_{G'+T^*}(v) \cap E(T)| \leq b(v)$ for $v \in V'$; (c) T contains exactly one of $v_e v_{u,e}$ and $v_e v_{u',e}$ for $e = uu' \in E(H)$; and (d) $E(T^*) \subseteq E(T)$.*

Proof. Since T is a spanning tree with maximum degree at most d, (a), (b), and (c) are obvious. By (b), $T - E(T^*)$ contains no path connecting two distinct components of $G' - \{v \in V' \mid b(v) = 1\}$. Since each connected component of $G' - \{v \in V' \mid b(v) = 1\}$ contains exactly one vertex in L, for any pair of vertices $v_1, v_2 \in L$, the unique v_1-v_2 path in T^* must be contained in T. This shows that $E(T^*) \subseteq E(T)$, because L is the set of all the leaves of T^*. ◀

For a spanning tree T in G with maximum degree at most d, we define an orientation σ_T of H as follows: an edge $e = uu' \in E(H)$ is inward for u if $v_e v_{u',e} \in E(T)$, and it is outward for u if $v_e v_{u,e} \in E(T)$. This defines an orientation of H by Lemma 9 (c). The following two lemmas show the correspondence between NCL configurations in H and spanning trees in G with maximum degree at most d.

▶ **Lemma 10.** *For any spanning tree T in G with maximum degree at most d, the orientation σ_T is an NCL configuration of H.*

Proof. It suffices to show that, for any $u \in V(H)$, the total weights of incoming arcs at u is at least two in σ_T.

First, let $u \in V(H)$ be an OR-vertex with $\delta_H(u) = \{e_1, e_2, e_3\}$. Since T is a spanning tree, it holds that $r_u v_{u,e_i} \in E(T)$ for some $i \in \{1, 2, 3\}$. Then, since $|\delta_{G'}(v_{u,e_i}) \cap E(T)| \leq b(v_{u,e_i}) - 1 = 1$ by (b) and (d) in Lemma 9, it holds that $v_{e_i} v_{u,e_i} \notin E(T)$. This means that e_i is inward for u in σ_T, and hence the total weights of incoming arcs at u is at least two.

Second, let $u \in V(H)$ be an AND-vertex with $\delta_H(u) = \{e_0, e_1, e_2\}$, where e_0 is a weight-2 edge and e_1 and e_2 are weight-1 edges. Since T is a spanning tree, we have either $r_u v_{u,e_0} \in E(T)$ or $r_u w_u \in E(T)$. If $r_u v_{u,e_0} \in E(T)$, then e_0 is inward for u in σ_T, which implies that the total weights of incoming arcs at u is at least two. Therefore, it suffices to consider the case when $r_u w_u \in E(T)$. Since $|\delta_{G'}(v) \cap E(T)| \leq 2$ for $v \in \{w_u, x_u, y_u, v_{u,e_1}, v_{u,e_2}\}$ by (b) and (d) in Lemma 9, we have either $\{r_u w_u, w_u x_u, x_u v_{u,e_1}, v_{u,e_1} v_{u,e_2}, v_{u,e_2} y_u\} \subseteq E(T)$ or $\{r_u w_u, w_u y_u, y_u v_{u,e_2}, v_{u,e_2} v_{u,e_1}, v_{u,e_1} x_u\} \subseteq E(T)$. In either case, $v_{e_i} v_{u,e_i} \notin E(T)$ for $i \in \{1, 2\}$, because $|\delta_{G'}(v_{u,e_i}) \cap E(T)| \leq 2$. This means that e_i is inward for u in σ_T for $i \in \{1, 2\}$, and hence the total weights of incoming arcs at u is at least two.

Therefore, σ_T is an NCL configuration of H. ◀

▶ **Lemma 11.** *For any NCL configuration σ of H, we can construct a spanning tree T in G with maximum degree at most d such that $\sigma_T = \sigma$ in polynomial time.*

Proof. Given an NCL configuration σ of H, we construct a spanning subgraph T of G such that

$$E(T) := E(T^*) \cup \{v\bar{v}_i \mid v \in V',\ i \in \{1, 2, \dots, d - b(v)\}\} \cup \{f_e \mid e \in E(H)\} \cup \bigcup_{u \in V(H)} F_u,$$

where an edge f_e for $e \in E(H)$ and an edge set F_u for $u \in V(H)$ are defined as follows.

- For an edge $e = uu' \in E(H)$, define $f_e := v_e v_{u',e}$ if e is inward for u in σ and define $f_e := v_e v_{u,e}$ otherwise.
- For an OR-vertex $u \in V(H)$ with $\delta_H(u) = \{e_1, e_2, e_3\}$, choose an arbitrarily edge e_i that is inward for u in σ and define $F_u := \{r_u v_{u,e_i}\}$. Note that such e_i exists as σ is an NCL configuration.
- For an AND-vertex $u \in V(H)$ with $\delta_H(u) = \{e_0, e_1, e_2\}$, where e_0 is a weight-2 edge and e_1 and e_2 are weight-1 edges, define $F_u = E'_u \setminus \{r_u w_u, v_{u,e_1} v_{u,e_2}\}$ if e_0 is inward for u in σ, and define $F_u := E'_u \setminus \{r_u v_{u,e_0}, w_u y_u\}$ otherwise.

Then, T is a spanning tree in G with maximum degree at most d such that $\sigma_T = \sigma$, which completes the proof. ◀

For two NCL configurations σ_s and σ_t of H, by Lemma 11, we can construct spanning trees T_s and T_t in G with maximum degree at most d such that $\sigma_{T_\mathsf{s}} = \sigma_\mathsf{s}$ and $\sigma_{T_\mathsf{t}} = \sigma_\mathsf{t}$. This yields an instance $(G, d, T_\mathsf{s}, T_\mathsf{t})$ of RST with Small Maximum Degree.

Correctness. In order to show the PSPACE-hardness of RST with Small Maximum Degree, we show that the original instance $(H, \sigma_\mathsf{s}, \sigma_\mathsf{t})$ of NCL Reconfiguration is equivalent to the obtained instance $(G, d, T_\mathsf{s}, T_\mathsf{t})$ of RST with Small Maximum Degree, that is, we prove that $(H, \sigma_\mathsf{s}, \sigma_\mathsf{t})$ is a YES-instance if and only if $(G, d, T_\mathsf{s}, T_\mathsf{t})$ is a YES-instance. To this end, we use the following lemma.

▶ **Lemma 12.** *Let T_1 and T_2 be spanning trees in G with maximum degree at most d. If σ_{T_1} and σ_{T_2} are adjacent, then there is a reconfiguration sequence from T_1 to T_2 in which all the spanning trees have maximum degree at most d.*

Proof. Let $e^* \in E(H)$ be the unique edge in H whose direction is different in σ_1 and σ_2. We prove the existence of a reconfiguration sequence by induction on $|E(T_1) \setminus E(T_2)|$. If $|E(T_1) \setminus E(T_2)| = 1$, then T_1 and T_2 are adjacent, and hence the claim is obvious. Suppose that $|E(T_1) \setminus E(T_2)| \geq 2$. Since σ_{T_1} and σ_{T_2} are adjacent, there exists a vertex $u \in V(H)$ such that T_1 and T_2 contain different edge sets in the gadget corresponding to u. That is, $(E(T_1) \setminus E(T_2)) \cap \delta_{G'}(r_u) \neq \emptyset$ for an OR-vertex $u \in V(H)$ or $(E(T_1) \setminus E(T_2)) \cap E'_u \neq \emptyset$ for an AND-vertex $u \in V(H)$. We fix such a vertex $u \in V(H)$.

Suppose that u is an OR-vertex such that $(E(T_1) \setminus E(T_2)) \cap \delta_{G'}(r_u) \neq \emptyset$. In this case, $E(T_1) \cap \delta_{G'}(r_u) = \{r_u v_{u,e_i}\}$ and $E(T_2) \cap \delta_{G'}(r_u) = \{r_u v_{u,e_j}\}$ for some distinct $i, j \in \{1, 2, 3\}$. By changing the roles of T_1 and T_2 if necessary, we may assume that either $e^* \notin \delta_H(u)$ or e^* is inward for u in σ_1. Then, $T'_1 := T_1 - r_u v_{u,e_i} + r_u v_{u,e_j}$ is a spanning tree with maximum degree at most d such that T'_1 is adjacent to T_1, $\sigma_{T'_1} = \sigma_{T_1}$, and $|E(T'_1) \setminus E(T_2)| = |E(T_1) \setminus E(T_2)| - 1$. By induction hypothesis, T'_1 is reconfigurable to T_2, and hence T_1 is reconfigurable to T_2.

Suppose that u is an AND-vertex such that $|(E(T_1) \setminus E(T_2)) \cap E'_u| = 1$. By changing the roles of T_1 and T_2 if necessary, we may assume that either $e^* \notin \delta_H(u)$ or e^* is inward for u in σ_{T_1}. Then, $T'_1 := T_1 - (E(T_1) \cap E'_u) + (E(T_2) \cap E'_u)$ is a spanning tree with maximum degree at most d such that T'_1 is adjacent to T_1, $\sigma_{T'_1} = \sigma_{T_1}$, and $|E(T'_1) \setminus E(T_2)| = |E(T_1) \setminus E(T_2)| - 1$. By induction hypothesis, T'_1 is reconfigurable to T_2, and hence T_1 is reconfigurable to T_2.

The remaining case is that u is an AND-vertex such that $|(E(T_1) \setminus E(T_2)) \cap E'_u| \geq 2$. Since each of T_1 and T_2 contains exactly one edge in $\delta_{G'}(r_u)$ and exactly four edges in $E'_u \setminus \delta_{G'}(r_u)$, we have that $|(E(T_1) \setminus E(T_2)) \cap \delta_{G'}(r_u)| = 1$ and $|(E(T_1) \setminus E(T_2)) \cap (E'_u \setminus \delta_{G'}(r_u))| = 1$. By changing the roles of T_1 and T_2 if necessary, we may assume that $E(T_1) \cap \delta_{G'}(r_u) = \{r_u v_{u,e_0}\}$ and $E(T_2) \cap \delta_{G'}(r_u) = \{r_u w_u\}$. This implies that $v_{u,e_0} v_{e_0} \notin E(T_1)$, and hence e_0 is inward for u in σ_{T_1}. We consider the following two cases separately.

- Suppose that e_0 is inward for u in σ_{T_2}. In this case, $T_2' := T_2 - r_u w_u + r_u v_{u,e_0}$ is a spanning tree with maximum degree at most d such that T_2' is adjacent to T_2, $\sigma_{T_2'} = \sigma_{T_2}$, and $|E(T_1) \setminus E(T_2')| = |E(T_1) \setminus E(T_2)| - 1$. By induction hypothesis, T_1 is reconfigurable to T_2', and hence T_1 is reconfigurable to T_2.
- Suppose that e_0 is outward for u in σ_{T_2}. In this case, e_1 and e_2 are inward for u in σ_{T_2} by Lemma 10. Furthermore, since $e^* = e_0$ holds, e_1 and e_2 are inward for u also in σ_{T_1}, that is, $v_{u,e_1}v_{e_1}, v_{u,e_2}v_{e_2} \notin E(T_1)$. Then, $T_1' := T_1 - (E(T_1) \cap (E_u' \setminus \delta_{G'}(r_u))) + (E(T_2) \cap (E_u' \setminus \delta_{G'}(r_u)))$ is a spanning tree with maximum degree at most d such that T_1' is adjacent to T_1, $\sigma_{T_1'} = \sigma_{T_1}$, and $|E(T_1') \setminus E(T_2)| = |E(T_1) \setminus E(T_2)| - 1$. By induction hypothesis, T_1' is reconfigurable to T_2, and hence T_1 is reconfigurable to T_2.

By the above argument, there is a reconfiguration sequence from T_1 to T_2. ◀

We are now ready to show the equivalence of $(H, \sigma_{\mathsf{s}}, \sigma_{\mathsf{t}})$ and $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$.

▶ **Lemma 13.** *Let $(H, \sigma_{\mathsf{s}}, \sigma_{\mathsf{t}})$ be an instance of* NCL Reconfiguration *and $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ be an instance of* RST with Small Maximum Degree *obtained by the above construction. Then, $(H, \sigma_{\mathsf{s}}, \sigma_{\mathsf{t}})$ is a* YES*-instance if and only if $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ is a* YES*-instance.*

Proof. We first show the "if" part. Suppose that there exists a reconfiguration sequence $\langle T_{\mathsf{s}} = T_0, T_1, \ldots, T_k = T_{\mathsf{t}} \rangle$ from T_{s} to T_{t}, where T_i is a spanning tree in G with maximum degree at most d for any $i \in \{0, 1, \ldots, k\}$ and T_i and T_{i+1} are adjacent for any $i \in \{0, 1, \ldots, k-1\}$. Then, σ_{T_i} is an NCL configuration of H for $i \in \{0, 1, \ldots, k\}$ by Lemma 10, and we have either $\sigma_{T_i} = \sigma_{T_{i+1}}$ or σ_{T_i} and $\sigma_{T_{i+1}}$ are adjacent for $i \in \{0, 1, \ldots, k-1\}$ as $|E(T_i) \setminus E(T_{i+1})| \leq 1$. Since $\sigma_{T_0} = \sigma_{T_{\mathsf{s}}} = \sigma_{\mathsf{s}}$ and $\sigma_{T_k} = \sigma_{T_{\mathsf{t}}} = \sigma_{\mathsf{t}}$, there exists a sequence of adjacent NCL configurations from σ_{s} to σ_{t}.

To show the "only-if" part, suppose that there exists a reconfiguration sequence $\langle \sigma_{\mathsf{s}} = \sigma_0, \sigma_1, \ldots, \sigma_k = \sigma_{\mathsf{t}} \rangle$, where σ_i is an NCL configuration of H for any $i \in \{0, 1, \ldots, k\}$ and σ_i and σ_{i+1} are adjacent for any $i \in \{0, 1, \ldots, k-1\}$. For $i \in \{1, 2, \ldots, k-1\}$, let T_i be a spanning tree in G with maximum degree at most d such that $\sigma_{T_i} = \sigma_i$, whose existence is guaranteed by Lemma 11. Let $T_0 := T_{\mathsf{s}}$ and $T_k := T_{\mathsf{t}}$. Since Lemma 12 shows that there is a reconfiguration sequence from T_i to T_{i+1} for $i \in \{0, 1, \ldots, k-1\}$, T_{s} is reconfigurable to T_{t}. ◀

This lemma shows that the above construction gives a polynomial reduction from NCL Reconfiguration to RST with Small Maximum Degree. Therefore, RST with Small Maximum Degree is PSPACE-hard, which completes the proof of Theorem 2.

4.2 A Solvable Special Case

In this subsection, we show a sufficient condition for the reconfigurability of instances. The condition is as follows; at least one of T_{s} and T_{t} has maximum degree at most $d-1$. Without loss of generality, we may assume that T_{t} satisfies the condition. Under this assumption, we have the following lemma.

▶ **Lemma 14.** *Suppose that $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ is an instance of* RST with Small Maximum Degree *such that T_{t} has maximum degree at most $d-1$. There exists an edge $e = xy \in E(T_{\mathsf{t}}) \setminus E(T_{\mathsf{s}})$ such that $d_{T_{\mathsf{s}}}(x) \leq d-1$ and $d_{T_{\mathsf{s}}}(y) \leq d-1$.*

Proof. To derive a contradiction, assume that Lemma 14 does not hold, that is, for any $e = xy \in E(T_\mathsf{t}) \setminus E(T_\mathsf{s})$, we have $d_{T_\mathsf{s}}(x) = d$ or $d_{T_\mathsf{s}}(y) = d$. Let $T^*_\mathsf{s} := T_\mathsf{s} - E(T_\mathsf{t})$ and $T^*_\mathsf{t} := T_\mathsf{t} - E(T_\mathsf{s})$. Note that $|E(T^*_\mathsf{s})| = |E(T^*_\mathsf{t})|$, because both T_s and T_t are spanning trees in G. Let $S := \{v \in V \mid d_{T_\mathsf{s}}(v) = d,\ d_{T^*_\mathsf{t}}(v) \geq 1\}$. With this notation, the assumption means that S forms a vertex cover of T^*_t. In what follows, we compare $|E(T^*_\mathsf{s})|$ and $|E(T^*_\mathsf{t})|$.

Let $X_1 := \{v \in V \mid d_{T^*_\mathsf{s}}(v) = 1\}$ and $X_{\geq 2} := \{v \in V \mid d_{T^*_\mathsf{s}}(v) \geq 2\}$. Then, we see that

$$\frac{1}{2}\sum_{v\in V} d_{T^*_\mathsf{s}}(v) = |E(T^*_\mathsf{s})| < |X_1 \cup X_{\geq 2}|, \tag{2}$$

because T^*_s is a forest. We also see that, for any $v \in S$,

$$d_{T^*_\mathsf{s}}(v) = d - |\delta_{T_\mathsf{s}}(v) \cap \delta_{T_\mathsf{t}}(v)| \geq d_{T_\mathsf{t}}(v) + 1 - |\delta_{T_\mathsf{s}}(v) \cap \delta_{T_\mathsf{t}}(v)| = d_{T^*_\mathsf{t}}(v) + 1 \tag{3}$$

holds, and hence $S \subseteq X_{\geq 2}$. With these observations, we obtain

$$\begin{aligned}
|E(T^*_\mathsf{s})| &= \sum_{v\in V} d_{T^*_\mathsf{s}}(v) - \frac{1}{2}\sum_{v\in V} d_{T^*_\mathsf{s}}(v) \\
&> \sum_{v\in V} d_{T^*_\mathsf{s}}(v) - |X_1 \cup X_{\geq 2}| && \text{(by (2))} \\
&= \sum_{v\in X_{\geq 2}} (d_{T^*_\mathsf{s}}(v) - 1) \\
&\geq \sum_{v\in S} (d_{T^*_\mathsf{s}}(v) - 1) && \text{(by } S \subseteq X_{\geq 2}\text{)} \\
&\geq \sum_{v\in S} d_{T^*_\mathsf{t}}(v) && \text{(by (3))} \\
&\geq |E(T^*_\mathsf{t})|, && \text{(because } S \text{ is a vertex cover of } T^*_\mathsf{t}\text{)}
\end{aligned}$$

which is a contradiction to $|E(T^*_\mathsf{s})| = |E(T^*_\mathsf{t})|$. Therefore, Lemma 14 holds. ◀

Let $e \in E(T_\mathsf{t}) \setminus E(T_\mathsf{s})$ be the edge as in the lemma and let $e' \in E(T_\mathsf{s}) \setminus E(T_\mathsf{t})$ be an edge such that $T'_\mathsf{s} := T_\mathsf{s} + e - e'$ is a spanning tree in G. Note that the maximum degree of T'_s is at most d. Since $|E(T'_\mathsf{s}) \setminus E(T_\mathsf{t})| = |E(T_\mathsf{s}) \setminus E(T_\mathsf{t})| - 1$, $(G, d, T'_\mathsf{s}, T_\mathsf{t})$ is a YES-instance by induction. This implies that T_s is reconfigurable to T_t, and thus the following theorem holds.

▶ **Theorem 15.** *If at least one of T_s and T_t has maximum degree at most $d - 1$, then an instance $(G, d, T_\mathsf{s}, T_\mathsf{t})$ of* RST WITH SMALL MAXIMUM DEGREE *is a* YES-*instance.*

Note that the above discussion shows that we can find a reconfiguration sequence $\langle T_\mathsf{s} = T_0, T_1, \ldots, T_k = T_\mathsf{t}\rangle$ with $k = |E(T_\mathsf{s}) \setminus E(T_\mathsf{t})|$, which is a shortest reconfiguration sequence, in polynomial time. Moreover, Theorem 15 implies the following corollary.

▶ **Corollary 16.** *Let G be a graph and d be a positive integer. If G contains a spanning tree with maximum degree at most $d - 1$, then any instance $(G, d, T_\mathsf{s}, T_\mathsf{t})$ of* RST WITH SMALL MAXIMUM DEGREE *is a* YES-*instance.*

Proof. Let T^* be a spanning tree in G with maximum degree at most $d-1$. Then, Theorem 15 shows that T_s is reconfigurable to T^* and T^* is reconfigurable to T_t. Hence, T_s is reconfigurable to T_t, which completes the proof. ◀

We note that it is not easy to determine whether or not G contains a spanning tree with maximum degree at most $d - 1$ even when $d = 3$, because finding a Hamiltonian path in cubic graphs is NP-hard.

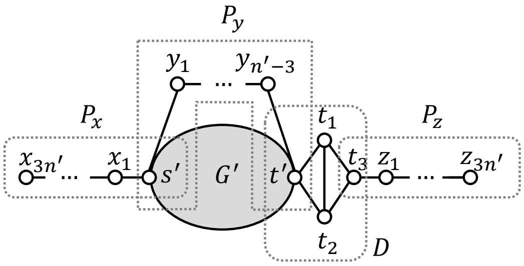


Figure 6 The graph G in the corresponding instance.

5 Large Diameter (Proof of Theorem 3)

In this section, we prove Theorem 3, which we restate here.

▶ **Theorem 3.** RST WITH LARGE DIAMETER *is NP-hard even restricted to planar graphs.*

To prove the theorem, we give a polynomial reduction from HAMILTONIAN PATH problem to RST WITH LARGE DIAMETER. A *Hamiltonian path* of a graph G is a path that visits each vertex of G exactly once. Given a graph $G = (V, E)$ and two vertices $s, t \in V$, the HAMILTONIAN PATH problem asks to determine whether or not G has a Hamiltonian path whose endpoints are s and t, which is known to be NP-hard [15].

Reduction. Let (G', s', t') be an instance of HAMILTONIAN PATH. We may assume that G' is connected, since otherwise (G', s', t') is trivially a NO-instance. We construct a corresponding instance $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ of RST WITH LARGE DIAMETER as follows (see Figure 6).

Let n' be the number of vertices in G', that is, $n' = |V(G')|$. We first add three vertices t_1, t_2, and t_3, and five edges $t't_1$, $t't_2$, t_1t_2, t_1t_3, and t_2t_3 to G'. Let D be the subgraph induced by $\{t', t_1, t_2, t_3\}$, which is isomorphic to the so-called diamond graph. We then add three paths $P_x = (x_{3n'}, x_{3n'-1}, \dots, x_1, s')$, $P_y = (s', y_1, y_2, \dots, y_{n'-3}, t')$, and $P_z = (t_3, z_1, z_2, \dots, z_{3n'})$, where all the vertices in P_x, P_y, and P_z except for s', t', and t_3 are distinct new vertices. Note that $|E(P_x)| = |E(P_z)| = 3n'$ and $|E(P_y)| = n' - 2$. Let $G = (V, E)$ be the obtained graph and set $d = 7n' + 1$.

Let F' be an arbitrary spanning forest in G' such that F' consists of two connected components (trees) of which one contains s' and the other contains t'. Then, define spanning trees T_{s} and T_{t} in G by

$$E(T_{\mathsf{s}}) = E(P_x) \cup E(P_y) \cup E(P_z) \cup E(F') \cup \{t't_1, t_1t_2, t_2t_3\},$$
$$E(T_{\mathsf{t}}) = E(P_x) \cup E(P_y) \cup E(P_z) \cup E(F') \cup \{t't_2, t_1t_2, t_1t_3\}.$$

We notice that $E(T_{\mathsf{s}}) \setminus E(F')$ forms a path in T_{s} of length $d = 7n' + 1$, and hence $\mathsf{diam}(T_{\mathsf{s}}) \geq d$. Similarly, $E(T_{\mathsf{t}}) \setminus E(F')$ forms a path in T_{t} of length d, and hence $\mathsf{diam}(T_{\mathsf{t}}) \geq d$. This completes the construction of the instance $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ of RST WITH LARGE DIAMETER.

Correctness. In the following, we show that G' contains a Hamiltonian path from s' to t' if and only if $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ is a YES-instance. The following lemma shows that the diameter of a spanning tree is dominated by the distance between $x_{3n'}$ and $z_{3n'}$.

▶ **Lemma 17** ($\star$). *For any spanning tree T in G, $\mathsf{diam}(T) = \bar{\ell}_T(x_{3n'}, z_{3n'})$.*

Thus, intuitively speaking, to keep the diameter and modify a spanning tree in D, we need to replace P_y with a slightly longer path in G'. Moreover, such a path must be a Hamiltonian path of G'. This observation yields the following lemma and completes the proof of Theorem 3.

▶ **Lemma 18** ($\star$)**.** *(G', s', t') is a* YES*-instance of* Hamiltonian Path *if and only if $(G, d, T_\mathsf{s}, T_\mathsf{t})$ is a* YES*-instance of* RST with Large Diameter*.*

6 Small Diameter (Proof of Theorem 4)

In this section, we prove Theorem 4, which we restate here.

▶ **Theorem 4.** RST with Small Diameter *is polynomial-time solvable.*

After giving some preliminaries for the proof in Section 6.1, we describe a naive algorithm for the problem in Section 6.2, which does not necessarily run in polynomial time. Then, by modifying it, we give a polynomial-time algorithm in Section 6.3.

6.1 Preliminaries for the Proof

Throughout the proof of Theorem 4, we fix a positive integer d. For each edge $e \in E$, we denote the middle point of e by p_e. We denote $R(H) := \{p_e \mid e \in E(H)\}$ for a subgraph H of G and let $R := R(G)$. Let $\hat{G}$ be the graph on $V \cup R$ that is obtained from G by subdividing each edge. Then, since $\bar{\ell}_G(u, v) = \frac{1}{2}\bar{\ell}_{\hat{G}}(u, v)$ for $u, v \in V$, we can naturally extend the domain of the distance to $V \cup R$ by setting $\bar{\ell}_G(u, v) := \frac{1}{2}\bar{\ell}_{\hat{G}}(u, v)$ for $u, v \in V \cup R$. We also define $\epsilon_G(v) := \max\left\{\bar{\ell}_G(v, u) \;\middle|\; u \in V\right\}$ for $v \in R$. If no confusion may arise, for $u, v \in V \cup R$, a u-v path in $\hat{G}$ is sometimes called a u-v path in G. We can see that spanning trees with diameter at most d are characterized as follows (see also [9]).

▶ **Lemma 19** ($\star$)**.** *For any spanning tree T in $G = (V, E)$, $\mathsf{diam}(T) \le d$ if and only if there exists $r \in V \cup R(T)$ such that $\epsilon_T(r) \le \frac{d}{2}$.*

We say that a subgraph Q of G is a *spanning pseudotree* if it is a connected spanning subgraph containing at most one cycle. In other words, a spanning pseudotree is obtained from a spanning tree by adding at most one edge. For brevity, a spanning pseudotree is simply called a *pseudotree*. For a pseudotree Q, let C_Q denote the unique cycle in Q if it exists. We can easily see that, for two spanning trees T_1 and T_2 with diameter at most d, $T_1 \leftrightarrow T_2$ if and only if $T_1 \cup T_2$ forms a pseudotree. For a pseudotree Q, we refer a point $r \in V \cup R(Q)$ as a *center point* of Q if $\epsilon_Q(r) \le \frac{d}{2}$. Note that a center point is not necessarily unique even if Q is a spanning tree. For a pseudotree Q, let $\mathsf{center}(Q) \subseteq V \cup R(Q)$ be the set of all center points of Q.

6.2 Algorithm Using Center Points: First Attempt

In this subsection, as a first step, we give an algorithm for RST with Small Diameter whose running time is not necessarily polynomial. In the same say as RST with Large Maximum Degree (Section 3), the proposed algorithm is based on testing the reachability in an auxiliary graph $\mathcal{G}$, which is defined as follows. The vertex set of $\mathcal{G}$ is defined as $V \cup R$, where each vertex v in $V(\mathcal{G})$ corresponds to the set of all the spanning trees containing v as a center point. For any pair u, v of distinct vertices in $V(\mathcal{G})$, there is an edge $uv \in E(\mathcal{G})$ if and only if there is a pseudotree Q with $u, v \in \mathsf{center}(Q)$. As we will see in Proposition 22

Algorithm 2 First algorithm for RST with Small Diameter.

```
Input: A graph $G$ and two spanning trees $T_{\mathsf{s}}$ and $T_{\mathsf{t}}$ in $G$ with diameter at most $d$.
Output: Is $T_{\mathsf{s}}$ reconfigurable to $T_{\mathsf{t}}$?
Compute $\mathsf{center}(T_{\mathsf{s}})$ and $\mathsf{center}(T_{\mathsf{t}})$, and construct $\mathcal{G}$;
if there is a path between $\mathsf{center}(T_{\mathsf{s}})$ and $\mathsf{center}(T_{\mathsf{t}})$ in $\mathcal{G}$ then return YES;
else return NO;
```

later, for two spanning trees T_u and T_v having center points u and v, respectively, $\mathcal{G}$ contains a u-v path if and only if T_u and T_v are reconfigurable to each other. Thus, to solve RST with Small Diameter, it is enough to find a path from a center point of T_{s} to a center point of T_{t} on $\mathcal{G}$. See Algorithm 2 for a pseudocode of our algorithm.

To show the correctness of Algorithm 2, we begin with easy but important lemmas.

▶ **Lemma 20** ($\star$)**.** *Let T_1 and T_2 be spanning trees in G with diameter at most d. If there exists a point $r \in \mathsf{center}(T_1) \cap \mathsf{center}(T_2)$, then T_1 is reconfigurable to T_2.*

▶ **Lemma 21** ($\star$)**.** *Let $r_1, r_2 \in V \cup R$ (possibly $r_1 = r_2$). There exists a pseudotree Q with $r_1, r_2 \in \mathsf{center}(Q)$ if and only if there exist two spanning trees T_1 and T_2 such that $r_i \in \mathsf{center}(T_i)$ for $i = 1, 2$ and $T_1 \leftrightarrow T_2$ (possibly $T_1 = T_2$).*

By these lemmas, we can show the correctness of Algorithm 2.

▶ **Proposition 22.** *Let T_{s} and T_{t} be spanning trees with diameter at most d. Then, T_{s} is reconfigurable to T_{t} if and only if $\mathcal{G}$ contains a path from $\mathsf{center}(T_{\mathsf{s}})$ to $\mathsf{center}(T_{\mathsf{t}})$.*

Proof. We first show the "only if" part. Suppose that there exists a reconfiguration sequence $\langle T_{\mathsf{s}} = T_0, T_1, \ldots, T_k = T_{\mathsf{t}} \rangle$ from T_{s} to T_{t}, where T_i is a spanning tree of diameter at most d for any $i \in \{0, 1, \ldots, k\}$ and $T_i \leftrightarrow T_{i+1}$ for any $i \in \{0, 1, \ldots, k-1\}$. Let r_i be a center point of T_i, where its existence is guaranteed by Lemma 19. For $i \in \{0, 1, \ldots, k-1\}$, by Lemma 21, there exists a pseudotree Q_i having both r_i and r_{i+1} as center points. This means that either $r_i = r_{i+1}$ or $\mathcal{G}$ contains an edge $r_i r_{i+1}$. Since $r_0 \in \mathsf{center}(T_{\mathsf{s}})$ and $r_k \in \mathsf{center}(T_{\mathsf{t}})$, $\mathcal{G}$ contains a path from $\mathsf{center}(T_{\mathsf{s}})$ to $\mathsf{center}(T_{\mathsf{t}})$.

To show the "if" part, suppose that $\mathcal{G}$ contains a path $(r_0, r_1, \ldots, r_k)$ from $\mathsf{center}(T_{\mathsf{s}})$ to $\mathsf{center}(T_{\mathsf{t}})$. For $i \in \{0, 1, \ldots, k-1\}$, since $r_i r_{i+1} \in E(\mathcal{G})$ implies the existence of a pseudotree Q_i with $r_i, r_{i+1} \in \mathsf{center}(Q_i)$, Lemma 21 shows that there exist two spanning trees T_i^+ and T_{i+1}^- such that $r_i \in \mathsf{center}(T_i^+)$, $r_{i+1} \in \mathsf{center}(T_{i+1}^-)$, and $T_i^+ \leftrightarrow T_{i+1}^-$. Let $T_0^- := T_{\mathsf{s}}$ and $T_k^+ := T_{\mathsf{t}}$. Then, for $i \in \{0, 1, \ldots, k\}$, since T_i^- and T_i^+ share r_i as a center point, T_i^- is reconfigurable to T_i^+ by Lemma 20. This together with $T_i^+ \leftrightarrow T_{i+1}^-$ shows that T_{s} is reconfigurable to T_{t}. ◀

Although this proposition shows the correctness of Algorithm 2, it does not imply a polynomial-time algorithm for RST with Small Diameter, because it is not easy to construct $\mathcal{G}$ efficiently. Indeed, for $u, v \in V(\mathcal{G})$, we do not know how to decide whether $uv \in E(\mathcal{G})$ or not in polynomial time. To avoid this problem, we efficiently construct a subgraph $\mathcal{G}'$ of $\mathcal{G}$ such that the reachability of $\mathcal{G}'$ is equal to that of $\mathcal{G}$, which is a key ingredient of our algorithm and discussed in the next subsection.

6.3 Modified Algorithm

In this subsection, we give a polynomial-time algorithm for RST with Small Diameter. In our algorithm, it is important to uniquely determine a shortest path between two points. To achieve this, we use a *perturbation technique* (see e.g., [4]).

For each edge e in G, we give a unique index $i(e) \in \{1, 2, \dots, |E|\}$ to e. For $j \in \{1, 2, \dots, |E|\}$, let $\chi_j \in \mathbb{R}^{|E|}$ be the unit vector such that the jth coordinate is one and the other coordinates are zero. For $e \in E$, define $\ell(e) \coloneqq (1, \chi_{i(e)}) \in \mathbb{R} \times \mathbb{R}^{|E|}$. For two vectors $x, y \in \mathbb{R}^k$, we denote $x < y$ if x is lexicographically smaller than y. For two paths P_1 and P_2 in G, we say that P_1 is *shorter than* P_2 if $\ell(P_1) := \sum_{e \in E(P_1)} \ell(e)$ is lexicographically smaller than $\ell(P_2) := \sum_{e \in E(P_2)} \ell(e)$. Since the first coordinate of $\ell(P_i)$ is $|E(P_i)|$ for $i = 1, 2$, if $|E(P_1)| < |E(P_2)|$, then P_1 is shorter than P_2. When $|E(P_1)| = |E(P_2)|$, we use the other coordinates to break ties. For $u, v \in V$, we define $\ell_G(u, v) := \min_P \sum_{e \in E(P)} \ell(e)$, where the minimum is taken over all the u-v paths P. Since $P_1 \neq P_2$ implies that $\ell(P_1) \neq \ell(P_2)$, the *shortest path* between two vertices is uniquely determined. We note that the unique shortest paths between two given vertices can be computed by using a standard shortest path algorithm. The running time is increased by the perturbation, but it is still polynomial.

For an edge $e = uv \in E$ of length $\ell(e) \in \mathbb{R} \times \mathbb{R}^{|E|}$, we regard e as a curve connecting u and v. An interior point p on e is represented by a triplet (u, v, α) with $\alpha \in \mathbb{R} \times \mathbb{R}^{|E|}$ such that $\mathbf{0} \leq \alpha \leq \ell(e)$, where $\leq$ means the lexicographical order. Here, α represents the length between u and p, and hence (u, v, α) and $(v, u, \ell(e) - \alpha)$ represent the same point. For two points $p_1 = (u_1, v_1, \alpha_1)$ and $p_2 = (u_2, v_2, \alpha_2)$ in G, consider a curve C connecting p_1 and p_2 that consists of a u_1-u_2 path P, a curve in u_1v_1 between u_1 and p_1, and a curve in u_2v_2 between u_2 and p_2. Such a curve C is called a *p_1-p_2 path* in G, and its length is defined as $\ell(C) := \sum_{e \in E(P)} \ell(e) + \alpha_1 + \alpha_2$.

For a point $r \in V \cup R$, the *shortest path tree from r* is the spanning tree in G that contains the unique shortest r-v path for any $v \in V$. For a pseudotree Q and for two points x and y on Q, let $Q[x, y]$ denote the shortest x-y path in Q, where we use this notation only when the shortest x-y path is uniquely determined. For $\alpha \in \mathbb{R} \times \mathbb{R}^{|E|}$, let $\bar{\alpha}$ denote the first coordinate of α, that is, $\bar{\alpha}$ is the length before the perturbation.

We denote $r_1 \xleftrightarrow{Q} r_2$ if Q is a pseudotree and $r_1, r_2 \in \mathsf{center}(Q)$ with $r_1 \neq r_2$. For any pseudotree Q and any points r_1 and r_2 with $r_1 \xleftrightarrow{Q} r_2$, we say that a triplet (r_1, r_2, Q) is *good* if

1. $\mathsf{label}_{r_1,r_2,Q}(v) \leq \mathsf{label}_{r_1,r_2,Q}(u) + \ell(uv)$ for any $uv \in E$, and
2. C_Q contains both r_1 and r_2 if C_Q exists,

where $\mathsf{label}_{r_1,r_2,Q}(v) := \max\{\ell_Q(r_1, v),\ \ell_Q(r_2, v)\}$. Roughly speaking, the first condition means that $\mathsf{label}_{r_1,r_2,Q}(v)$ can be seen as the distance from a certain point to v in an auxiliary graph. If r_1 and r_2 are clear from the context, $\mathsf{label}_{r_1,r_2,Q}(v)$ is simply denoted by $\mathsf{label}_Q(v)$. We define the graph $\mathcal{G}'$ as follows: $V(\mathcal{G}') = V \cup R$ and $\mathcal{G}'$ contains an edge r_1r_2 if and only if there is a pseudotree Q such that $r_1 \xleftrightarrow{Q} r_2$ and (r_1, r_2, Q) is good. Clearly, $\mathcal{G}'$ is a subgraph of $\mathcal{G}$.

The following theorem shows that we can determine whether $r_1r_2 \in E(\mathcal{G}')$ or not in polynomial time, whose proof is given in the full version.

▶ **Theorem 23** ($\star\star$). *Let r_1 and r_2 be points in $V \cup R$ with $r_1 \neq r_2$. We can find in polynomial time a pseudotree Q such that $r_1 \xleftrightarrow{Q} r_2$ and (r_1, r_2, Q) is good if it exists.*

The next theorem shows that the reachability of $\mathcal{G}'$ is equal to that of $\mathcal{G}$, which is a key property of $\mathcal{G}'$. A proof is given in the full version.

▶ **Theorem 24** ($\star\star$). *For any $r_1, r_2 \in V \cup R$ with $r_1r_2 \in E(\mathcal{G})$, $\mathcal{G}'$ contains an r_1-r_2 path.*

We are now ready to prove Theorem 4. By Proposition 22 and Theorem 24, two spanning trees T_s and T_t are reconfigurable to each other if and only if $\mathcal{G}'$ contains a path from $\mathsf{center}(T_\mathsf{s})$ to $\mathsf{center}(T_\mathsf{t})$. Since we can construct $\mathcal{G}'$ in polynomial time by Theorem 23, this

Algorithm 3 Modified algorithm for RST WITH SMALL DIAMETER.

```
Input: A graph $G$ and two spanning trees $T_s$ and $T_t$ in $G$ with diameter at most $d$.
Output: Is $T_s$ reconfigurable to $T_t$?
Compute center($T_s$) and center($T_t$), and construct $\mathcal{G}' = (V \cup R, \emptyset)$;
for $r_1, r_2 \in V \cup R$ with $r_1 \neq r_2$ do
    if there is a pseudotree $Q$ such that $r_1 \overset{Q}{\longleftrightarrow} r_2$ and $(r_1, r_2, Q)$ is good then
        Add an edge $r_1 r_2$ to $\mathcal{G}'$;
if there is a path between center($T_s$) and center($T_t$) in $\mathcal{G}'$ then return YES;
else return NO;
```

can be tested in polynomial time. Therefore, RST WITH SMALL DIAMETER can be solved in polynomial time, which completes the proof of Theorem 4. A pseudocode of our algorithm is given in Algorithm 3. Note that all the proofs are constructive, and hence we can find a desired reconfiguration sequence from T_s to T_t in polynomial time if it exists.

7 Concluding Remarks

In this paper, we have investigated the computational complexity of RST WITH SMALL (or LARGE) MAXIMUM DEGREE and RST WITH SMALL (or LARGE) DIAMETER.

We have proved in Theorem 2 that RST WITH SMALL MAXIMUM DEGREE is PSPACE-complete for $d \geq 3$. One can naturally ask what happens for the case of maximum degree at most 2. In this case, the problem becomes the HAMILTONIAN PATH RECONFIGURATION problem, in which a feasible solution is a Hamiltonian path. We were not able to determine the complexity of this problem and we left it as an open problem. Note that HAMILTONIAN PATH RECONFIGURATION problem can be also seen as a special case of RST WITH LARGE DIAMETER in which the lower bound on the diameter is $|V(G)| - 1$. Note also that, for the Hamiltonian *cycle* case, the HAMILTONIAN CYCLE RECONFIGURATION problem is known to be PSPACE-complete [20], in which two edge flips are executed in one step.

We have proved in Theorem 3 that RST WITH LARGE DIAMETER is NP-hard, but it is unclear whether this problem belongs to the class NP. We conjecture that the problem is PSPACE-complete, and left this question as another open problem.

References

1 Matthieu Barjon, Arnaud Casteigts, Serge Chaumette, Colette Johnen, and Yessin M. Neggaz. Maintaining a spanning forest in highly dynamic networks: The synchronous case. In Marcos K. Aguilera, Leonardo Querzoni, and Marc Shapiro, editors, *Principles of Distributed Systems*, pages 277–292, Cham, 2014. Springer International Publishing.

2 Nicolas Bousquet, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Paul Ouvrard, Akira Suzuki, and Kunihiro Wasa. Reconfiguration of spanning trees with many or few leaves. In *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, pages 24:1–24:15, 2020. doi:10.4230/LIPIcs.ESA.2020.24.

3 Nicolas Bousquet, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Paul Ouvrard, Akira Suzuki, and Kunihiro Wasa. Reconfiguration of spanning trees with degree constraint or diameter constraint, 2022. arXiv:2201.04354.

4 Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-source shortest paths in embedded graphs. *SIAM Journal on Computing*, 42(4):1542–1571, 2013. doi:10.1137/120864271.

5 Artur Czumaj and Willy-B. Strothmann. Bounded degree spanning trees. In Rainer Burkard and Gerhard Woeginger, editors, *Algorithms — ESA '97*, volume 1284 of *LNCS*, pages 104–117, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

6 Martin Furer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994. doi:10.1006/jagm.1994.1042.

7 Michel X. Goemans. Minimum bounded degree spanning trees. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 273–282. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.48.

8 Refael Hassin and Asaf Levin. Minimum restricted diameter spanning trees. *Discrete Applied Mathematics*, 137(3):343–357, 2004. doi:10.1016/S0166-218X(03)00360-3.

9 Refael Hassin and Arie Tamir. On the minimum diameter spanning tree problem. *Inf. Process. Lett.*, 53(2):109–111, January 1995. doi:10.1016/0020-0190(94)00183-Y.

10 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005. doi:10.1016/j.tcs.2005.05.008.

11 Jan van den Heuvel. The complexity of change. In Simon R. Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

12 Silu Huang, Ada Wai-Chee Fu, and Ruifeng Liu. Minimum spanning trees in temporal graphs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 419–430, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2723372.2723717.

13 Giuseppe F. Italiano and Rajiv Ramaswami. Maintaining spanning trees of small diameter. *Algorithmica*, 22(3):275–304, 1998. doi:10.1007/PL00009225.

14 Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

15 Richard M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

16 Haruka Mizuta, Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. Reconfiguration of minimum steiner trees via vertex exchanges. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany.*, pages 79:1–79:11, 2019. doi:10.4230/LIPIcs.MFCS.2019.79.

17 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

18 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *J. ACM*, 62(1), March 2015. doi:10.1145/2629366.

19 Michael J. Spriggs, J. Mark Keil, Sergei Bespamyatnikh, Michael Segal, and Jack Snoeyink. Computing a $(1+\varepsilon)$-approximate geometric minimum-diameter spanning tree. *Algorithmica*, 38(4):577–589, 2004. doi:10.1007/s00453-003-1056-z.

20 Asahi Takaoka. Complexity of Hamiltonian cycle reconfiguration. *Algorithms*, 11(9), 2018. doi:10.3390/a11090140.

A Proofs for Section 3 (Large Maximum Degree (Proof of Theorem 1))

▶ **Lemma 6** ($\star$). *Let T_1 and T_2 be spanning trees in G with maximum degree at least d. If there exists a vertex $u \in \mathsf{large}(T_1) \cap \mathsf{large}(T_2)$, then T_1 is reconfigurable to T_2.*

Proof. We show that T_1 is reconfigurable to T_2 by induction on $d - |\delta_{T_1}(u) \cap \delta_{T_2}(u)|$.

Suppose that $d - |\delta_{T_1}(u) \cap \delta_{T_2}(u)| \leq 0$ holds. By Lemma 5, there exists a reconfiguration sequence from T_1 to T_2 in which all the spanning trees contain $\delta_{T_1}(u) \cap \delta_{T_2}(u)$. This shows that, for any spanning tree T' in the sequence, $|\delta_{T'}(u)| \geq |\delta_{T_1}(u) \cap \delta_{T_2}(u)| \geq d$. Hence, T_1 is reconfigurable to T_2.

Suppose that $d - |\delta_{T_1}(u) \cap \delta_{T_2}(u)| \geq 1$ holds. Since $|\delta_{T_2}(u)| \geq d$ and $|\delta_{T_1}(u) \cap \delta_{T_2}(u)| \leq d - 1$, there exists an edge $e \in \delta_{T_2}(u) \setminus \delta_{T_1}(u)$. Since $T_1 + e$ contains a unique cycle C and T_2 contains no cycle, there exists an edge $f \in E(C) \setminus E(T_2)$. Then, we have that $f \in E(T_1) \setminus E(T_2)$ and $T_1' := T_1 + e - f$ is a spanning tree in G. Observe that $|\delta_{T_1'}(u)| \geq |\delta_{T_1}(u) \cup \{e\}| - 1 \geq |\delta_{T_1}(u)| \geq d$, which shows that $u \in \mathsf{large}(T_1')$. We also see that $d - |\delta_{T_1'}(u) \cap \delta_{T_2}(u)| = d - |\delta_{T_1}(u) \cap \delta_{T_2}(u)| - 1$. Therefore, by the induction hypothesis, T_1' is reconfigurable to T_2. This shows that T_1 is reconfigurable to T_2 as T_1 and T_1' are adjacent. ◀

▶ **Lemma 7** ($\star$). *Let T_{s} and T_{t} be spanning trees with maximum degree at least d. Then, T_{s} is reconfigurable to T_{t} if and only if $\mathcal{G}$ contains a path from $\mathsf{large}(T_{\mathsf{s}})$ to $\mathsf{large}(T_{\mathsf{t}})$.*

Proof. We first show the "only if" part. Suppose that there exists a reconfiguration sequence $\langle T_{\mathsf{s}} = T_0, T_1, \ldots, T_k = T_{\mathsf{t}} \rangle$ from T_{s} to T_{t}, where T_i is a spanning tree of maximum degree at least d for any $i \in \{0, 1, \ldots, k\}$ and T_i and T_{i+1} are adjacent for any $i \in \{0, 1, \ldots, k-1\}$. For each i, let v_i be a vertex in $\mathsf{large}(T_i)$. By the definition of $\mathcal{G}$, for $i \in \{0, 1, \ldots, k-1\}$, we have either $v_i = v_{i+1}$ or $\mathcal{G}$ contains an edge $v_i v_{i+1}$. Since $v_0 \in \mathsf{large}(T_{\mathsf{s}})$ and $v_k \in \mathsf{large}(T_{\mathsf{t}})$, $\mathcal{G}$ contains a path from $\mathsf{large}(T_{\mathsf{s}})$ to $\mathsf{large}(T_{\mathsf{t}})$.

To show the "if" part, suppose that $\mathcal{G}$ contains a path $(v_0, v_1, \ldots, v_k)$ from $\mathsf{large}(T_{\mathsf{s}})$ to $\mathsf{large}(T_{\mathsf{t}})$. For $i \in \{0, 1, \ldots, k-1\}$, $v_i v_{i+1} \in E(\mathcal{G})$ means that there exist two spanning trees T_i^+ and T_{i+1}^- such that $v_i \in \mathsf{large}(T_i^+)$, $v_{i+1} \in \mathsf{large}(T_{i+1}^-)$, and $T_i^+ \leftrightarrow T_{i+1}^-$. Let $T_0^- := T_{\mathsf{s}}$ and $T_k^+ := T_{\mathsf{t}}$. Then, for $i \in \{0, 1, \ldots, k\}$, since $v_i \in \mathsf{large}(T_i^-) \cap \mathsf{large}(T_i^+)$, T_i^- is reconfigurable to T_i^+ by Lemma 6. This together with $T_i^+ \leftrightarrow T_{i+1}^-$ shows that T_{s} is reconfigurable to T_{t}. ◀

▶ **Lemma 8** ($\star$). *For two distinct vertices $u, v \in V$, there exists an edge $uv \in E(\mathcal{G})$ if and only if $|N_G(u)| \geq d$, $|N_G(v)| \geq d$, and*

$$|N_G(u) \cup N_G(v)| \geq \begin{cases} 2d - 1 & \text{if } uv \in E(G), \\ 2d - 2 & \text{otherwise.} \end{cases} \tag{1}$$

Proof. We first prove the "only-if" direction. Suppose that $\mathcal{G}$ contains an edge uv, that is, there exist spanning trees T and T' such that $u \in \mathsf{large}(T)$, $v \in \mathsf{large}(T')$, and $T \leftrightarrow T'$ (possibly $T = T'$). Then, $|N_G(u)| \geq d$ and $|N_G(v)| \geq d$ are obvious. Since T contains no cycle, we know that $N_T(u)$ and $N_T(v)$ contain at most one common vertex. Then, we obtain

$$\begin{aligned} |N_G(u) \cup N_G(v)| &\geq |N_T(u) \cup N_T(v)| \\ &\geq |N_T(u)| + |N_T(v)| - 1 && \text{(by } |N_T(u) \cap N_T(v)| \leq 1\text{)} \\ &\geq |N_T(u)| + (|N_{T'}(v)| - 1) - 1 && \text{(by } |E(T') \setminus E(T)| \leq 1\text{)} \\ &\geq 2d - 2. && (4) \end{aligned}$$

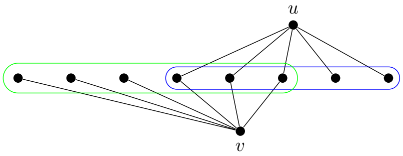


Figure 7 Case when $uv \notin E(G)$.

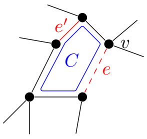


Figure 8 Cycle C and edge e'.

Similarly, if $uv \in E(G) \setminus E(T)$, then we obtain

$$|N_G(u) \cup N_G(v)| \geq |N_T(u) \cup N_T(v) \cup \{u, v\}| \geq |N_T(u)| + |N_T(v)| + 1 \geq 2d. \tag{5}$$

If $uv \in E(T)$, then $N_T(u) \cap N_T(v) = \emptyset$ holds, and hence we obtain

$$|N_G(u) \cup N_G(v)| \geq |N_T(u) \cup N_T(v)| = |N_T(u)| + |N_T(v)| \geq 2d - 1. \tag{6}$$

By (4), (5), and (6), we obtain (1).

We next prove the "if" direction. Suppose that $|N_G(u)| \geq d$, $|N_G(v)| \geq d$, and (1) hold. For each of the following two cases, we define an edge set $F \subseteq E$.

- Suppose that $uv \notin E(G)$ holds (Figure 7). Let $S_u \subseteq N_G(u)$ be a vertex set with $|S_u| = d$ that maximizes $|S_u \setminus N_G(v)|$. Then, we have either $S_u \subseteq N_G(u) \setminus N_G(v)$ or $S_u \supsetneq N_G(u) \setminus N_G(v)$. If $S_u \subseteq N_G(u) \setminus N_G(v)$, then let $S_v \subseteq N_G(v)$ be a vertex set with $|S_v| = d - 1$. Otherwise, let $S_v \subseteq N_G(v)$ be a vertex set such that $|S_v| = d - 1$ and $|S_u \cap S_v| = 1$, where such S_v exists because $|N_G(v) \setminus S_u| = |(N_G(u) \cup N_G(v)) \setminus S_u| \geq d - 2$ and $|N_G(v) \cap S_u| \geq 1$. In either case, we obtain $S_u \subseteq N_G(u)$ and $S_v \subseteq N_G(v)$ such that $|S_u| = d$, $|S_v| = d - 1$, and $|S_u \cap S_v| \leq 1$. Define $F := \{uw \mid w \in S_u\} \cup \{vw \mid w \in S_v\}$.
- Suppose that $uv \in E(G)$ holds. Since $|N_G(u) \setminus \{v\}| \geq d - 1$, $|N_G(v) \setminus \{u\}| \geq d - 1$, and $|(N_G(u) \setminus \{v\}) \cup (N_G(v) \setminus \{u\})| \geq 2d - 3$, by the same argument as above, we can take $S_u \subseteq N_G(u) \setminus \{v\}$ and $S_v \subseteq N_G(v) \setminus \{u\}$ such that $|S_u| = d - 1$, $|S_v| = d - 2$, and $S_u \cap S_v = \emptyset$. Define $F := \{uw \mid w \in S_u\} \cup \{vw \mid w \in S_v\} \cup \{uv\}$.

In both cases, it holds that $|F \cap \delta_G(u)| = d$, $|F \cap \delta_G(v)| = d - 1$, and F contains no cycle. Therefore, there exists a spanning tree T with $E(T) \supseteq F$ such that $|\delta_T(u)| \geq |F \cap \delta_G(u)| = d$ and $|\delta_T(v)| \geq |F \cap \delta_G(v)| = d - 1$. If $|\delta_T(v)| \geq d$, then we obtain $\{u, v\} \subseteq \mathsf{large}(T)$, which shows that $uv \in E(\mathcal{G})$. Therefore, it suffices to consider the case when $|\delta_T(v)| = d - 1$. Since $|\delta_G(v)| \geq d$, there exists an edge $e \in \delta_G(v) \setminus \delta_T(v)$. Let C be the unique cycle in $T + e$ and e' be an edge in $E(C) \setminus \delta_T(v)$ (see Figure 8). Then, $T' := T + e - e'$ is a spanning tree such that $|\delta_{T'}(v)| = |\delta_T(v) \cup \{e\}| = d$, which means that $v \in \mathsf{large}(T')$. Since T and T' are adjacent, we obtain $uv \in E(\mathcal{G})$. ◀

B Proofs for Section 5 (Large Diameter (Proof of Theorem 3))

▶ **Lemma 17** ($\star$). *For any spanning tree T in G,* $\mathsf{diam}(T) = \bar{\ell}_T(x_{3n'}, z_{3n'})$.

Proof. Let T be a spanning tree in G and P^* be a longest path in T. For $x, y \in V$ and for a spanning tree T in G, we denote by $T[x, y]$ the unique path between x and y in T. Since $T[x_{3n'}, z_{3n'}]$ contains all the edges in P_x and P_z, the length of $T[x_{3n'}, z_{3n'}]$ is at least $6n'$, and hence $|E(P^*)| \geq |E(T[x_{3n'}, z_{3n'}])| \geq 6n'$. Since each of $G - \{x_1, \dots, x_{3n'}\}$ and $G - \{z_1, \dots, z_{3n'}\}$ contains at most $5n'$ vertices, we obtain $V(P^*) \cap \{x_1, \dots, x_{3n'}\} \neq \emptyset$ and $V(P^*) \cap \{z_1, \dots, z_{3n'}\} \neq \emptyset$. This shows that $P^* = T[x_i, z_j]$ for some $i, j \in \{1, 2, \dots, 3n'\}$. Since $T[x_i, z_j]$ is a subpath of $T[x_{3n'}, z_{3n'}]$, P^* must be equal to $T[x_{3n'}, z_{3n'}]$, that is, $\mathsf{diam}(T) = \bar{\ell}_T(x_{3n'}, z_{3n'})$. ◀

▶ **Lemma 18** (⋆). *(G', s', t') is a* YES*-instance of* Hamiltonian Path *if and only if $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ is a* YES*-instance of* RST with Large Diameter*.*

Proof. We first prove the "if" direction. Suppose that $(G, d, T_{\mathsf{s}}, T_{\mathsf{t}})$ is a YES-instance. Then there is a reconfiguration sequence $\langle T_{\mathsf{s}} = T_0, T_1, \ldots, T_k = T_{\mathsf{t}}\rangle$ between T_{s} and T_{t} in which all the spanning trees have diameter at least d. Let T_i be the first spanning tree in the sequence such that T_i is obtained from T_{i-1} by exchanging an edge in D, that is, $E(T_j) \cap E(D) = \{t't_1, t_1t_2, t_2t_3\}$ for all $j \in \{0, 1, \ldots, i-1\}$ and $E(T_i) \cap E(D) \neq \{t't_1, t_1t_2, t_2t_3\}$. Note that such i exists, because $E(T_k) \cap E(D) \neq \{t't_1, t_1t_2, t_2t_3\}$. Note also that $\bar{\ell}_{T_i}(t', t_3) = 2$ by the definition of T_i. Then, by Lemma 17, we obtain

$$\begin{aligned}7n' + 1 &\leq \mathsf{diam}(T_i)\\ &= \bar{\ell}_{T_i}(x_{3n'}, z_{3n'})\\ &= \bar{\ell}_{T_i}(x_{3n'}, s') + \bar{\ell}_{T_i}(s', t') + \bar{\ell}_{T_i}(t', t_3) + \bar{\ell}_{T_i}(t_3, z_{3n'})\\ &= 3n' + \bar{\ell}_{T_i}(s', t') + 2 + 3n',\end{aligned}$$

and hence $\bar{\ell}_{T_i}(s', t') \geq n' - 1$. Since P_y contains only $n' - 2$ edges, all the edges in $T_i[s', t']$ are contained in G'. We thus conclude that $T_i[s', t']$ is a Hamiltonian path between s' and t' in G', and hence the "if" direction follows.

We now prove the "only-if" direction. Suppose that (G', s', t') is a YES-instance, that is, G' contains a Hamiltonian path P^* between s' and t'. Let e^* be any edge in P^* and e_y be any edge in P_y. We define five spanning trees T_1, T_2, T_3, T_4, and T_5 in G as follows:

$$\begin{aligned}E(T_1) &= E(P_x) \cup E(P_y) \cup E(P_z) \cup E(P^* - e^*) \cup \{t't_1, t_1t_2, t_2t_3\},\\ E(T_2) &= E(P_x) \cup E(P_y - e_y) \cup E(P_z) \cup E(P^*) \cup \{t't_1, t_1t_2, t_2t_3\},\\ E(T_3) &= E(P_x) \cup E(P_y - e_y) \cup E(P_z) \cup E(P^*) \cup \{t't_2, t_1t_2, t_2t_3\},\\ E(T_4) &= E(P_x) \cup E(P_y - e_y) \cup E(P_z) \cup E(P^*) \cup \{t't_2, t_1t_2, t_1t_3\},\\ E(T_5) &= E(P_x) \cup E(P_y) \cup E(P_z) \cup E(P^* - e^*) \cup \{t't_2, t_1t_2, t_1t_3\}.\end{aligned}$$

We observe that $\langle T_1, T_2, T_3, T_4, T_5\rangle$ is a reconfiguration sequence from T_1 and T_5 in which all the spanning trees have diameter at least $d = 7n' + 1$. Thus, in order to show that T_{s} is reconfigurable to T_{t}, it suffices to show that T_{s} is reconfigurable to T_1 and T_5 is reconfigurable to T_{t}. Since $T_{\mathsf{s}}[x_{3n'}, z_{3n'}] = T_1[x_{3n'}, z_{3n'}]$, Lemma 5 shows that there is a reconfiguration sequence from T_{s} to T_1 in which all the spanning trees contain $E(T_{\mathsf{s}}[x_{3n'}, z_{3n'}]) \subseteq E(T_{\mathsf{s}}) \cap E(T_1)$. Therefore, every spanning tree in the sequence has diameter at least d, and hence T_{s} is reconfigurable to T_1. Similarly, T_5 is reconfigurable to T_{t}. By combining them, we have that T_{s} is reconfigurable to T_{t}, which completes the proof of the "only-if" direction. ◀

C Proofs for Section 6 (Small Diameter (Proof of Theorem 4))

▶ **Lemma 19** (⋆). *For any spanning tree T in $G = (V, E)$, $\mathsf{diam}(T) \leq d$ if and only if there exists $r \in V \cup R(T)$ such that $\epsilon_T(r) \leq \frac{d}{2}$.*

Proof. To show the "if" part, suppose that there exists $r \in V \cup R(T)$ such that $\epsilon_T(r) \leq \frac{d}{2}$. Then, for any $u, v \in V$, $\bar{\ell}_T(u, v) \leq \bar{\ell}_T(u, r) + \bar{\ell}_T(r, v) \leq 2\epsilon_T(r) \leq d$, which shows that $\mathsf{diam}(T) \leq d$.

To show the "only-if" part, suppose that $\mathsf{diam}(T) \leq d$. Let $d^* := \mathsf{diam}(T)$ and let $u, v \in V$ be a pair of vertices such that $\bar{\ell}_T(u, v) = d^*$. Let $r \in V \cup R(T)$ be the middle point of u and v in T, that is, $\bar{\ell}_T(u, r) = \bar{\ell}_T(r, v) = \frac{d^*}{2}$. Since T is a spanning tree, for any $x \in V$, $\frac{d^*}{2} + \bar{\ell}_T(r, x) = \max\{\bar{\ell}_T(u, x),\ \bar{\ell}_T(v, x)\} \leq d^*$. This shows that $\bar{\ell}_T(r, x) \leq \frac{d^*}{2}$, that is, $\epsilon_T(r) \leq \frac{d}{2}$. ◀

▶ **Lemma 20** (⋆). *Let T_1 and T_2 be spanning trees in G with diameter at most d. If there exists a point $r \in \mathsf{center}(T_1) \cap \mathsf{center}(T_2)$, then T_1 is reconfigurable to T_2.*

Proof. Let T^* be the spanning tree that is obtained by applying the breadth first search from r in G. Here, if $r \in R$ is the middle point of $uv \in E$, then the breadth first search is started from $\{u, v\}$. Since $\bar{\ell}_{T^*}(r, v) \leq \bar{\ell}_{T_1}(r, v) \leq \frac{d}{2}$ for any $v \in V$, the diameter of T^* is at most d. For $v \in V$, let $P_{T^*}(v)$ (resp. $P_{T_1}(v)$) denote the unique path from r to v in T^* (resp. T_1). In T^*, we say that a vertex $u \in V$ is the *parent* of v if $uv \in E(T^*)$ and $\bar{\ell}_{T^*}(r, v) = \bar{\ell}_{T^*}(r, u) + 1$. The parent in T_1 is defined in the same way.

In order to show that T_1 is reconfigurable to T_2, it suffices to show that T_i is reconfigurable to T^* for $i \in \{1, 2\}$. Suppose that $T_1 \neq T^*$ and let xy be an edge in $E(T^*) \setminus E(T_1)$ that minimizes $\min\{\bar{\ell}_{T^*}(r, x), \bar{\ell}_{T^*}(r, y)\}$. Without loss of generality, we assume that x is the parent of y in T^*. Let $w \in V$ be the parent of y in T_1 and define $T_1' := T_1 + \{xy\} - \{wy\}$, which is a spanning tree in G. By the choice of xy, we obtain $P_{T_1}(x) = P_{T^*}(x)$, and hence $P_{T_1'}(y) = P_{T^*}(y)$ and $\bar{\ell}_{T_1'}(r, y) = \bar{\ell}_{T^*}(r, y) \leq \bar{\ell}_{T_1}(r, y)$. Since this shows that $\bar{\ell}_{T_1'}(r, v) \leq \bar{\ell}_{T_1}(r, v) \leq \frac{d}{2}$ for any $v \in V$, the diameter of T_1' is at most d by Lemma 19. We observe that replacing T_1 with T_1' increases $|\{v \in V \mid P_{T_1}(v) = P_{T^*}(v)\}|$ by at least one, because $P_{T_1}(y) \neq P_{T_1'}(y) = P_{T^*}(y)$. Therefore, by applying this procedure at most $|V|$ times, we obtain a reconfiguration sequence from T_1 to T^*. We can also obtain a reconfiguration sequence from T_2 to T^* in the same way. Hence, the statement holds. ◀

▶ **Lemma 21** (⋆). *Let $r_1, r_2 \in V \cup R$ (possibly $r_1 = r_2$). There exists a pseudotree Q with $r_1, r_2 \in \mathsf{center}(Q)$ if and only if there exist two spanning trees T_1 and T_2 such that $r_i \in \mathsf{center}(T_i)$ for $i = 1, 2$ and $T_1 \leftrightarrow T_2$ (possibly $T_1 = T_2$).*

Proof. We first consider the "if" part. Suppose that there exist two spanning trees T_1 and T_2 such that $r_i \in \mathsf{center}(T_i)$ for $i = 1, 2$ and $T_1 \leftrightarrow T_2$. Then $Q := T_1 \cup T_2$ is a desired pseudotree as $\epsilon_Q(r_i) \leq \epsilon_{T_i}(r_i) \leq \frac{d}{2}$ for $i = 1, 2$.

We next consider the "only-if" part. Suppose that Q is a pseudotree with $r_1, r_2 \in \mathsf{center}(Q)$. For $i = 1, 2$, let T_i be the spanning tree that is obtained by applying the breadth first search from r_i in Q. Then, we obtain $\epsilon_{T_i}(r_i) = \epsilon_Q(r_i) \leq \frac{d}{2}$. Furthermore, since $|E(T_1) \setminus E(T_2)| \leq |E(Q) \setminus E(T_2)| = 1$, it holds that $T_1 \leftrightarrow T_2$. ◀

Characterizing Omega-Regularity Through Finite-Memory Determinacy of Games on Infinite Graphs

Patricia Bouyer
Université Paris-Saclay, CNRS, ENS Paris-Saclay,
Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Mickael Randour
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Pierre Vandenhove
F.R.S.-FNRS & UMONS – Université de Mons, Belgium
Université Paris-Saclay, CNRS, ENS Paris-Saclay,
Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Abstract

We consider zero-sum games on infinite graphs, with objectives specified as sets of infinite words over some alphabet of *colors*. A well-studied class of objectives is the one of ω*-regular objectives*, due to its relation to many natural problems in theoretical computer science. We focus on the strategy complexity question: given an objective, how much memory does each player require to play as well as possible? A classical result is that finite-memory strategies suffice for both players when the objective is ω-regular. We show a reciprocal of that statement: when both players can play optimally with a *chromatic* finite-memory structure (i.e., whose updates can only observe colors) in all infinite game graphs, then the objective must be ω-regular. This provides a game-theoretic characterization of ω-regular objectives, and this characterization can help in obtaining memory bounds. Moreover, a by-product of our characterization is a new *one-to-two-player lift*: to show that chromatic finite-memory structures suffice to play optimally in two-player games on infinite graphs, it suffices to show it in the simpler case of one-player games on infinite graphs. We illustrate our results with the family of discounted-sum objectives, for which ω-regularity depends on the value of some parameters.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases two-player games on graphs, infinite arenas, finite-memory determinacy, optimal strategies, ω-regular languages

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.16

Related Version *Full Version*: `https://arxiv.org/abs/2110.01276` [9]

Funding This work has been partially supported by the ANR Project MAVeriQ (ANR-ANR-20-CE25-0012).
Mickael Randour: Mickael Randour is an F.R.S.-FNRS Research Associate.
Pierre Vandenhove: Pierre Vandenhove is an F.R.S.-FNRS Research Fellow.

1 Introduction

Games on graphs and synthesis. We study *zero-sum turn-based games on infinite graphs*. In such games, two players, $\mathcal{P}_1$ and $\mathcal{P}_2$, interact for an infinite duration on a graph, called an *arena*, whose state space is partitioned into states controlled by $\mathcal{P}_1$ and states controlled by $\mathcal{P}_2$. The game starts in some state of the arena, and the player controlling the current state may choose the next state following an edge of the arena. Moves of the players in the game are prescribed by their *strategy*, which can use information about the past of the play. Edges of the arena are labeled with a (possibly infinite) alphabet of *colors*, and the interaction of the players in the arena generates an *infinite word* over this alphabet of colors. These infinite words can be used to specify the players' objectives: a *winning condition* is a set of infinite words, and $\mathcal{P}_1$ wins a game on a graph if the infinite word generated by its interaction with $\mathcal{P}_2$ on the arena belongs to this winning condition – otherwise, $\mathcal{P}_2$ wins.

This game-theoretic model has applications to the *reactive synthesis* problem [4]: a system (modeled as $\mathcal{P}_1$) wants to guarantee some specification (the winning condition) against an uncontrollable environment (modeled as $\mathcal{P}_2$). Finding a *winning strategy* in the game for $\mathcal{P}_1$ corresponds to building a controller for the system that achieves the specification against all possible behaviors of the environment.

Strategy complexity. We are interested in the *strategy complexity* question: given a winning condition, how *complex* must winning strategies be, and how *simple* can they be? We are interested in establishing the sufficient and necessary amount of memory to play *optimally*. We consider in this work that an *optimal strategy* in an arena must be winning from any state from which winning is possible (a property sometimes called *uniformity* in the literature). The amount of memory relates to how much information about the past is needed to play in an optimal way. With regard to reactive synthesis, this has an impact in practice on the resources required for an optimal controller.

Three classes of strategies are often distinguished, depending on the number of states of memory they use: memoryless, finite-memory, and infinite-memory strategies. A notable subclass of finite-memory strategies is the class of strategies that can be implemented with finite-memory structures that only observe the sequences of colors (and not the sequences of states nor edges). Such memory structures are called *chromatic* [30]. By contrast, finite-memory structures that have access to the states and edges of arenas are called *general*. Chromatic memory structures are syntactically less powerful and may require more states than general ones [11], but they have the benefit that they can be defined independently of arenas.

We seek to characterize the winning conditions for which chromatic-finite-memory strategies suffice to play optimally against arbitrarily complex strategies, for both players, in all finite and infinite arenas. We call this property *chromatic-finite-memory determinacy*. This property generalizes *memoryless determinacy*, which describes winning conditions for which memoryless strategies suffice to play optimally for both players in all arenas. Our work follows a line of research [6, 8] giving various characterizations of chromatic-finite-memory determinacy for games on *finite* arenas (see Remark 2 for more details).

ω-regular languages. A class of winning conditions commonly arising as natural specifications for reactive systems (it encompasses, e.g., linear temporal logic specifications [38]) consists of the *ω-regular languages*. They are, among other characterizations, the languages of infinite words that can be described by a *finite parity automaton* [36]. It is known that all ω-regular languages are chromatic-finite-memory determined, which is due to the facts that an ω-regular language is expressible with a parity automaton, and that *parity conditions* admit memoryless optimal strategies [27, 42]. Multiple works study the strategy complexity of ω-regular languages, giving, e.g., precise general memory requirements for all Muller conditions [18] or a characterization of the chromatic memory requirements of Muller conditions [11, Theorem 28].

A result in the other direction is given by Colcombet and Niwiński [17]: they showed that if a *prefix-independent* winning condition is memoryless-determined in infinite arenas, then this winning condition must be a parity condition. As parity conditions are memoryless-determined, this provides an elegant characterization of parity conditions from a strategic perspective, under prefix-independence assumption.

Congruence. A well-known tool to study a language L of finite (resp. infinite) words is its *right congruence relation* $\sim_L$: for two finite words w_1 and w_2, we write $w_1 \sim_L w_2$ if for all finite (resp. infinite) words w, $w_1 w \in L$ if and only if $w_2 w \in L$. There is a natural deterministic (potentially infinite) automaton recognizing the equivalence classes of the right congruence, called the *minimal-state automaton of* $\sim_L$ [41, 35].

The relation between a regular language of *finite* words and its right congruence is given by the Myhill-Nerode theorem [37], which provides a natural bijection between the states of the minimal deterministic automaton recognizing a regular language and the equivalence classes of its right congruence relation. Consequences of this theorem are that a language is regular if and only if its right congruence has finitely many equivalence classes, and a regular language can be recognized by the minimal-state automaton of its right congruence.

For the theory of languages of *infinite* words, the situation is not so simple: ω-regular languages have a right congruence with finitely many equivalence classes, but having finitely many equivalence classes does not guarantee ω-regularity (for example, a language is *prefix-independent* if and only if its right congruence has exactly one equivalence class, but this does not imply ω-regularity). Moreover, ω-regular languages cannot necessarily be recognized by adding a natural acceptance condition (parity, Rabin, Muller…) to the minimal-state automaton of their right congruence [1]. There has been multiple works about the links between a language of infinite words and the minimal-state automaton of its right congruence; one relevant question is to understand when a language can be recognized by this minimal-state automaton [41, 35, 1].

Contributions. We characterize the ω-regularity of a language of infinite words W through the strategy complexity of the zero-sum turn-based games on infinite graphs with winning condition W: the ω-regular languages are *exactly* the chromatic-finite-memory determined languages (seen as winning conditions) (Theorem 9). As discussed earlier, it is well-known that ω-regular languages admit chromatic-finite-memory optimal strategies [36, 42, 11] – our results yield the other implication. This therefore provides a characterization of ω-regular languages through a game-theoretic and strategic lens.

Our technical arguments consist in providing a precise connection between the representation of W as a parity automaton and a chromatic memory structure sufficient to play optimally. If strategies based on a chromatic finite-memory structure are sufficient to play optimally for both players, then W is recognized by a parity automaton built on top of the direct product of the *minimal-state automaton of the right congruence* and this *chromatic memory structure* (Theorem 8). This result generalizes the work from Colcombet and Niwiński [17] in two ways: by relaxing the prefix-independence assumption about the winning condition, and by generalizing the class of strategies considered from memoryless to chromatic-finite-memory strategies. We recover their result as a special case.

Moreover, we actually show that chromatic-finite-memory determinacy in *one-player* games of both players is sufficient to show ω-regularity of a language. As ω-regular languages are chromatic-finite-memory determined in two-player games, we can reduce the problem of chromatic-finite-memory determinacy of a winning condition in *two-player* games to the easier chromatic-finite-memory determinacy in *one-player* games (Theorem 10). Such a *one-to-two-player lift* holds in multiple classes of zero-sum games, such as deterministic games on finite arenas [23, 6, 31] and stochastic games on finite arenas [24, 8]. The proofs for finite arenas all rely on an *edge-induction technique* (also used in other works about strategy complexity in finite arenas [28, 21, 13]) that appears unfit to deal with infinite arenas. Although not mentioned by Colcombet and Niwiński, it was already noticed [30] that for prefix-independent winning conditions in games on infinite graphs, a one-to-two-player lift for *memoryless* determinacy follows from [17].

Related works. We have already mentioned [18, 42, 17, 29, 11] for fundamental results on the memory requirements of ω-regular conditions, [23, 24, 6, 8] for characterizations of "low" memory requirements in finite (deterministic and stochastic) arenas, and [41, 35, 1] for links between an ω-regular language and the minimal-state automaton of its right congruence.

One stance of our work is that our assumptions about strategy complexity affect *both* players. Another intriguing question is to understand when the memory requirements of only *one* player are finite. In finite arenas, results in this direction are sufficient conditions for the existence of memoryless optimal strategies for one player [28, 3], and a procedure to compute the chromatic memory requirements of prefix-independent ω-regular conditions [29, 30].

Other articles study the strategy complexity of (non-necessarily ω-regular) winning conditions in infinite arenas; see, e.g., [20, 25, 16]. In such non-ω-regular examples, as can be expected given our main result, at least one player needs infinite memory to play optimally, or the arena model is different from ours (e.g., only allowing finite branching – we discuss such differences in more depth after Theorem 8). A particularly interesting example w.r.t. our results is considered by Chatterjee and Fijalkow [15]. They study the strategy complexity of *finitary Büchi and parity conditions*, and show that $\mathcal{P}_1$ has chromatic-finite-memory optimal strategies for finitary Büchi and finitary parity. However, for these (non-ω-regular) winning conditions, $\mathcal{P}_2$ needs infinite memory. This example illustrates that our main result would not hold if we just focused on the strategy complexity of one player.

We mention works on finite-memory determinacy in different contexts: finite arenas [34], non-zero-sum games [33], countable one-player stochastic games [26], concurrent games [32, 7].

Structure. We fix definitions in Section 2. Our main results are discussed in Section 3. We apply our results to discounted-sum and mean-payoff winning conditions in Section 4. Due to a lack of space, we only sketch some technical details; the complete proofs as well as additional examples and remarks are found in the full version of the article [9].

2 Preliminaries

Let C be an arbitrary non-empty set of *colors*. Given a set A, we write A^* for the set of finite sequences of elements of A and A^ω for the set of infinite sequences of elements of A.

Arenas. We consider two players $\mathcal{P}_1$ and $\mathcal{P}_2$. An arena is a tuple $\mathcal{A} = (S, S_1, S_2, E)$ such that $S = S_1 \uplus S_2$ (disjoint union) is a non-empty set of *states* (of any cardinality) and $E \subseteq S \times C \times S$ is a set of *edges*. States in S_1 are controlled by $\mathcal{P}_1$ and states in S_2 are controlled by $\mathcal{P}_2$. We allow arenas with infinite branching. Given $e \in E$, we denote by in, col, and out the projections to its first, second, and third component, respectively (i.e., $e = (\mathsf{in}(e), \mathsf{col}(e), \mathsf{out}(e))$). We assume arenas to be *non-blocking*: for all $s \in S$, there exists $e \in E$ such that $\mathsf{in}(e) = s$.

Let $\mathcal{A} = (S, S_1, S_2, E)$ be an arena with $s \in S$. We denote by $\mathsf{Plays}(\mathcal{A}, s)$ the set of *plays of* $\mathcal{A}$ *from* s, that is, infinite sequences of edges $\rho = e_1e_2 \ldots \in E^\omega$ such that $\mathsf{in}(e_1) = s$ and for all $i \geq 1$, $\mathsf{out}(e_i) = \mathsf{in}(e_{i+1})$. For $\rho = e_1e_2 \ldots \in \mathsf{Plays}(\mathcal{A}, s)$, we write $\mathsf{col}^\omega(\rho)$ for the infinite sequence $\mathsf{col}(e_1)\mathsf{col}(e_2) \ldots \in C^\omega$. We denote by $\mathsf{Hists}(\mathcal{A}, s)$ the set of *histories of* $\mathcal{A}$ *from* s, which are all finite prefixes of plays of $\mathcal{A}$ from s. We write $\mathsf{Plays}(\mathcal{A})$ and $\mathsf{Hists}(\mathcal{A})$ for the sets of all plays of $\mathcal{A}$ and all histories of $\mathcal{A}$ (from any state), respectively. If $h = e_1 \ldots e_k$ is a history of $\mathcal{A}$, we define $\mathsf{in}(h) = \mathsf{in}(e_1)$ and $\mathsf{out}(h) = \mathsf{out}(e_k)$. For convenience, for every $s \in S$, we also consider the *empty history* λ_s from s, and we set $\mathsf{in}(\lambda_s) = \mathsf{out}(\lambda_s) = s$. For $i \in \{1, 2\}$, we denote by $\mathsf{Hists}_i(\mathcal{A})$ the set of histories h such that $\mathsf{out}(h) \in S_i$. An arena $\mathcal{A} = (S, S_1, S_2, E)$ is a *one-player arena of* $\mathcal{P}_1$ (resp. *of* $\mathcal{P}_2$) if $S_2 = \emptyset$ (resp. $S_1 = \emptyset$).

Skeletons. A *skeleton* is a tuple $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ such that M is a finite set of *states*, $m_{\mathsf{init}} \in M$ is an *initial state*, and $\alpha_{\mathsf{upd}} \colon M \times C \to M$ is an *update function.* We denote by α^*_{upd} the natural extension of α_{upd} to finite sequences of colors. We always assume that all states of skeletons are reachable from their initial state. We define the trivial skeleton $\mathcal{M}_{\mathsf{triv}}$ as the only skeleton with a single state. Although we require skeletons to have finitely many states, we allow them to have infinitely many transitions (which happens when C is infinite).

We say that a non-empty sequence $\pi = (m_1, c_1) \ldots (m_k, c_k) \in (M \times C)^+$ is a *path of* $\mathcal{M}$ *(from* m_1 *to* $\alpha_{\mathsf{upd}}(m_k, c_k)$*)* if for all $i \in \{1, \ldots, k-1\}$, $\alpha_{\mathsf{upd}}(m_i, c_i) = m_{i+1}$. For convenience, we also consider every element $(m, \perp)$ for $m \in M$ and $\perp \notin C$ to be an *empty path of* $\mathcal{M}$ *(from* m *to* m*)*. A non-empty path of $\mathcal{M}$ from m to m' is a *cycle of* $\mathcal{M}$ *(on* m*)* if $m = m'$. Cycles of $\mathcal{M}$ are usually denoted by letter γ. For $\pi = (m_1, c_1) \ldots (m_k, c_k)$ a path of $\mathcal{M}$, we define $\mathsf{col}^*(\pi)$ to be the sequence $c_1 \ldots c_k \in C^*$. For an infinite sequence $(m_1, c_1)(m_2, c_2) \ldots \in (M \times C)^\omega$, we also write $\mathsf{col}^\omega((m_1, c_1)(m_2, c_2) \ldots)$ for the infinite sequence $c_1c_2 \ldots \in C^\omega$.

For $m, m' \in M$, we write $\Pi_{m,m'}$ for the set of paths of $\mathcal{M}$ from m to m', Γ_m for the set of cycles of $\mathcal{M}$ on m, and $\Gamma_{\mathcal{M}}$ for the set of all cycles of $\mathcal{M}$ (on any skeleton state). When considering sets of paths or cycles of $\mathcal{M}$, we add a c in front of the set to denote the projections of the corresponding paths or cycles to colors (e.g., $\mathsf{c}\Gamma_{\mathcal{M}} = \{\mathsf{col}^*(\gamma) \in C^+ \mid \gamma \in \Gamma_{\mathcal{M}}\}$).

For $w = c_1c_2 \ldots \in C^\omega$, we define $\mathsf{skel}(w)$ as the infinite sequence $(m_1, c_1)(m_2, c_2) \ldots \in (M \times C)^\omega$ that w induces in the skeleton ($m_1 = m_{\mathsf{init}}$ and for all $i \geq 1$, $\alpha_{\mathsf{upd}}(m_i, c_i) = m_{i+1}$).

Let $\mathcal{M}_1 = (M_1, m^1_{\mathsf{init}}, \alpha^1_{\mathsf{upd}})$ and $\mathcal{M}_2 = (M_2, m^2_{\mathsf{init}}, \alpha^2_{\mathsf{upd}})$ be two skeletons. Their *(direct) product* $\mathcal{M}_1 \otimes \mathcal{M}_2$ is the skeleton $(M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ where $M = M_1 \times M_2$, $m_{\mathsf{init}} = (m^1_{\mathsf{init}}, m^2_{\mathsf{init}})$, and, for all $m_1 \in M_1$, $m_2 \in M_2$, $c \in C$, $\alpha_{\mathsf{upd}}((m_1, m_2), c) = (\alpha^1_{\mathsf{upd}}(m_1, c), \alpha^2_{\mathsf{upd}}(m_2, c))$.

Strategies. Let $\mathcal{A} = (S, S_1, S_2, E)$ be an arena and $i \in \{1, 2\}$. A *strategy of* $\mathcal{P}_i$ *on* $\mathcal{A}$ is a function $\sigma_i \colon \mathsf{Hists}_i(\mathcal{A}) \to E$ such that for all $h \in \mathsf{Hists}_i(\mathcal{A})$, $\mathsf{out}(h) = \mathsf{in}(\sigma_i(h))$. We denote by $\Sigma_i(\mathcal{A})$ the set of strategies of $\mathcal{P}_i$ on $\mathcal{A}$. Given a strategy σ_i of $\mathcal{P}_i$, we say that a play ρ is *consistent with* σ_i if for all finite prefixes $h = e_1 \ldots e_i$ of ρ such that $\mathsf{out}(h) \in S_i$, $\sigma_i(h) = e_{i+1}$. For $s \in S$, we denote by $\mathsf{Plays}(\mathcal{A}, s, \sigma_i)$ the set of plays from s that are consistent with σ_i.

For $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ a skeleton, a strategy $\sigma_i \in \Sigma_i(\mathcal{A})$ is *based on (memory)* $\mathcal{M}$ if there exists a function $\alpha_{\mathsf{nxt}} \colon S \times M \to E$ such that for all $s \in S_i$, $\sigma_i(\lambda_s) = \alpha_{\mathsf{nxt}}(s, m_{\mathsf{init}})$, and for all non-empty paths $h \in \mathsf{Hists}_i(\mathcal{A})$, $\sigma_i(h) = \alpha_{\mathsf{nxt}}(\mathsf{out}(h), \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, \mathsf{col}^*(h)))$. A strategy is *memoryless* if it is based on $\mathcal{M}_{\mathsf{triv}}$.

▶ Remark 1. Our memory model is *chromatic* [30], i.e., it observes the sequences of colors and not the sequences of edges of arenas, since the argument of the update function of a skeleton is in $M \times C$. It was recently shown that the amount of memory states required to play optimally for a winning condition using chromatic skeletons may be strictly larger than using *general* memory structures (i.e., using memory structures observing edges) [11, Proposition 32]. The example provided is a Muller condition (hence an ω-regular condition), in which both kinds of memory requirements are still finite. A result in this direction is also provided by Le Roux [32] for games on *finite* arenas: it shows that in many games, a strategy using general finite memory can be swapped for a (larger) chromatic finite memory.

For games on infinite arenas, which we consider in this article, we do not know whether there exists a winning condition with *finite* general memory requirements, but *infinite* chromatic memory requirements. Our results focus on chromatic memory requirements. ⌟

Winning conditions. A *(winning) condition* is a set $W \subseteq C^\omega$. When a condition W is clear in the context, we say that an infinite word $w \in C^\omega$ is winning if $w \in W$, and losing if not. For a condition W and a word $w \in C^*$, we write $w^{-1}W = \{w' \in C^\omega \mid ww' \in W\}$ for the set of *winning continuations of* w. We write $\overline{W}$ for the complement $C^\omega \setminus W$ of a condition W.

A *game* is a tuple $\mathcal{G} = (\mathcal{A}, W)$ where $\mathcal{A}$ is an arena and W is a winning condition.

Optimality and determinacy. Let $\mathcal{G} = (\mathcal{A} = (S, S_1, S_2, E), W)$ be a game, and $s \in S$. We say that $\sigma_1 \in \Sigma_1(\mathcal{A})$ *is winning from* s if $\mathsf{col}^\omega(\mathsf{Plays}(\mathcal{A}, s, \sigma_1)) \subseteq W$, and we say that $\sigma_2 \in \Sigma_2(\mathcal{A})$ *is winning from* s if $\mathsf{col}^\omega(\mathsf{Plays}(\mathcal{A}, s, \sigma_2)) \subseteq \overline{W}$.

A strategy of $\mathcal{P}_i$ is *optimal in* $(\mathcal{A}, W)$ if it is winning from all the states from which $\mathcal{P}_i$ has a winning strategy. We often write *optimal in* $\mathcal{A}$ if condition W is clear from the context. We stress that this notion of optimality requires a *single* strategy to be winning from *all* the winning states (a property sometimes called *uniformity*).

A winning condition W is *determined* if for all games $\mathcal{G} = (\mathcal{A} = (S, S_1, S_2, E), W)$, for all $s \in S$, either $\mathcal{P}_1$ or $\mathcal{P}_2$ has a winning strategy from s. Let $\mathcal{M}$ be a skeleton. We say that a winning condition W is $\mathcal{M}$*-determined* if (i) W is determined and (ii) in all arenas $\mathcal{A}$, both players have an optimal strategy based on $\mathcal{M}$. A winning condition W is *one-player* $\mathcal{M}$*-determined* if in all one-player arenas $\mathcal{A}$ of $\mathcal{P}_1$, $\mathcal{P}_1$ has an optimal strategy based on $\mathcal{M}$ *and* in all one-player arenas $\mathcal{A}$ of $\mathcal{P}_2$, $\mathcal{P}_2$ has an optimal strategy based on $\mathcal{M}$. A winning condition W is (one-player) *memoryless-determined* if it is (one-player) $\mathcal{M}_{\mathsf{triv}}$-determined. A winning condition W is *(one-player) chromatic-finite-memory determined* if there exists a skeleton $\mathcal{M}$ such that it is (one-player) $\mathcal{M}$-determined.

▶ Remark 2. It might seem surprising that for chromatic-finite-memory determinacy, we require the existence of a *single* skeleton that suffices to play optimally in *all* arenas, rather than the seemingly weaker existence, for each arena, of a finite skeleton (which may depend on the arena) that suffices to play optimally. In infinite arenas, it turns out that these notions are equivalent (proof in [9]).

▶ Lemma 3. *Let $W \subseteq C^\omega$ be a winning condition. The following are equivalent:*

1. *for all arenas $\mathcal{A}$, there exists a skeleton $\mathcal{M}^{\mathcal{A}}$ such that both players have an optimal strategy based on $\mathcal{M}^{\mathcal{A}}$ in $\mathcal{A}$;*
2. *W is chromatic-finite-memory determined.*

When restricted to finite arenas, we do not have an equivalence between these two notions (hence the distinction between finite-memory determinacy and *arena-independent* finite-memory determinacy [6, 8]). Our proof of Lemma 3 exploits that an infinite "union" of arenas is still an arena, which is not true when restricted to finite arenas. ⌟

ω-regular languages. We define a *parity automaton* as a pair $(\mathcal{M}, p)$ where $\mathcal{M}$ is a skeleton and $p\colon M \times C \to \{0, \dots, n\}$; function p assigns *priorities* to every transition of $\mathcal{M}$. This definition implies that we consider deterministic and complete parity automata (i.e., in every state, reading a color leads to exactly one state). Following [12], if $\mathcal{M}$ is a skeleton, we say that a parity automaton $(\mathcal{M}', p)$ is *defined on top of* $\mathcal{M}$ if $\mathcal{M}' = \mathcal{M}$.

A parity automaton $(\mathcal{M}, p)$ defines a language $L_{(\mathcal{M},p)}$ of all the infinite words $w \in C^\omega$ such that, for $\mathsf{skel}(w) = (m_1, c_1)(m_2, c_2)\ldots$, $\limsup_{i\geq 1} p(m_i, c_i)$ is even. We say that $W \subseteq C^\omega$ is *recognized by* $(\mathcal{M}, p)$ if $W = L_{(\mathcal{M},p)}$. A language of infinite words is *ω-regular* if it is recognized by a parity automaton. We emphasize that we consider *transition-based* parity conditions: we assign priorities to transitions (and not states) of $\mathcal{M}$. For more information on links between state-based and transition-based acceptance conditions, we refer to [11].

Right congruence. For $\sim$ an equivalence relation, we call the *index of* $\sim$ the number of equivalence classes of $\sim$. We denote by $[a]_\sim$ the equivalence class of an element a for $\sim$.

Let W be a winning condition. We define the *right congruence* $\sim_W \subseteq C^* \times C^*$ *of* W as $w_1 \sim_W w_2$ if $w_1^{-1}W = w_2^{-1}W$ (meaning that w_1 and w_2 have the same winning continuations). Relation $\sim_W$ is an equivalence relation. When W is clear from the context, we write $\sim$ for $\sim_W$. We denote by ε the empty word. When $\sim$ has finite index, we can associate a natural skeleton $\mathcal{M}_\sim = (M_\sim, m^\sim_{\mathsf{init}}, \alpha^\sim_{\mathsf{upd}})$ to $\sim$ such that $M_\sim$ is the set of equivalence classes of $\sim$, $m^\sim_{\mathsf{init}} = [\varepsilon]_\sim$, and $\alpha^\sim_{\mathsf{upd}}([w]_\sim, c) = [wc]_\sim$. This transition function is well-defined since it follows from the definition of $\sim$ that if $w_1 \sim w_2$, then for all $c \in C$, $w_1 c \sim w_2 c$. Hence, the choice of representatives for the equivalence classes does not have an impact in this definition. We call skeleton $\mathcal{M}_\sim$ the *minimal-state automaton of* $\sim$ [41, 35].

3 Concepts and characterization

We define two concepts at the core of our characterization, one of them dealing with *prefixes* and the other one dealing with *cycles*. Let $W \subseteq C^\omega$ be a winning condition and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a skeleton.

Prefix-independence. Let $\sim$ be the right congruence of W.

▶ **Definition 4.** *Condition W is $\mathcal{M}$-*prefix-independent *if for all $m \in M$, for all $w_1, w_2 \in \mathsf{c}\Pi_{m_{\mathsf{init}},m}$, $w_1 \sim w_2$.*

In other words, W is $\mathcal{M}$-prefix-independent if finite words reaching the same state of $\mathcal{M}$ from its initial state have the same winning continuations. The classical notion of *prefix-independence* is equivalent to $\mathcal{M}_{\mathsf{triv}}$-prefix-independence (as all finite words have the exact same set of winning continuations, which is W). If $\sim$ has finite index, W is in particular $\mathcal{M}_\sim$-prefix-independent: indeed, two finite words reach the same state of $\mathcal{M}_\sim$ (if and) only if they are equivalent for $\sim$. Any skeleton $\mathcal{M}$ such that W is $\mathcal{M}$-prefix-independent must have at least one state for each equivalence class of $\sim$, but multiple states may partition the same equivalence class.

Cycle-consistency. For $w \in C^*$, we define

$$\Gamma^{\mathsf{win},w}_{\mathcal{M}} = \{\gamma \in \Gamma_m \mid m = \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w) \text{ and } (\mathsf{col}^*(\gamma))^\omega \in w^{-1}W\}$$

as the cycles on the skeleton state reached by w in $\mathcal{M}$ that induce winning words when repeated infinitely many times after w. We define

$$\Gamma^{\mathsf{lose},w}_{\mathcal{M}} = \{\gamma \in \Gamma_m \mid m = \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w) \text{ and } (\mathsf{col}^*(\gamma))^\omega \in w^{-1}\overline{W}\}$$

as their losing counterparts. We emphasize that cycles are allowed to go through the same edge multiple times.

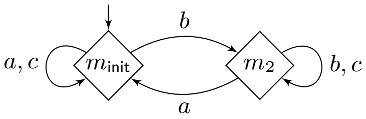


Figure 1 Skeleton $\mathcal{M}$ such that $W = \mathsf{Büchi}(a) \cap \mathsf{Büchi}(b)$ is $\mathcal{M}$-cycle-consistent (Example 6). In figures, we use rhombuses (resp. circles, squares) to depict skeleton states (resp. arena states controlled by $\mathcal{P}_1$, arena states controlled by $\mathcal{P}_2$).

▶ **Definition 5.** *Condition W is $\mathcal{M}$-*cycle-consistent *if for all $w \in C^*$, $(\mathsf{c}\Gamma_{\mathcal{M}}^{\mathsf{win},w})^\omega \subseteq w^{-1}W$ and $(\mathsf{c}\Gamma_{\mathcal{M}}^{\mathsf{lose},w})^\omega \subseteq w^{-1}\overline{W}$.*

What this says is that after any finite word, if we concatenate infinitely many winning (resp. losing) cycles on the skeleton state reached by that word, then it only produces winning (resp. losing) infinite words.

▶ **Example 6.** For $c' \in C$, let $\mathsf{Büchi}(c')$ be the set of infinite words on C that see color c' infinitely often. Let $C = \{a, b, c\}$. Condition $W = \mathsf{Büchi}(a) \cap \mathsf{Büchi}(b)$ is $\mathcal{M}_{\mathsf{triv}}$-prefix-independent, but not $\mathcal{M}_{\mathsf{triv}}$-cycle-consistent: for any $w \in C^*$, a and b are both in $\mathsf{c}\Gamma_{\mathcal{M}_{\mathsf{triv}}}^{\mathsf{lose},w}$ (as wa^ω and wb^ω are losing), but word $w(ab)^\omega$ is winning. However, W is $\mathcal{M}$-cycle-consistent for the skeleton $\mathcal{M}$ with two states m_{init} and m_2 represented in Figure 1. For finite words reaching m_{init}, the losing cycles only see a and c, and combining infinitely many of them gives an infinite word without b, which is a losing continuation of any finite word. The winning cycles are the ones that go to m_2 and then go back to m_{init}, as they must see both a and b; combining infinitely many of them guarantees a winning continuation after any finite word. A similar reasoning applies to state m_2. Notice that W is also $\mathcal{M}$-prefix-independent. With regard to memory requirements, condition W is not $\mathcal{M}_{\mathsf{triv}}$-determined but is $\mathcal{M}$-determined. ⌟

Both $\mathcal{M}$-prefix-independence and $\mathcal{M}$-cycle-consistency hold symmetrically for a winning condition and its complement, and are stable by product with an arbitrary skeleton (as products generate even smaller sets of prefixes and cycles to consider).

▶ **Lemma 7.** *Let $W \subseteq C^\omega$ be a winning condition and $\mathcal{M}$ be a skeleton. Then, W is $\mathcal{M}$-prefix-independent (resp. $\mathcal{M}$-cycle-consistent) if and only if $\overline{W}$ is $\mathcal{M}$-prefix-independent (resp. $\mathcal{M}$-cycle-consistent). If W is $\mathcal{M}$-prefix-independent (resp. $\mathcal{M}$-cycle-consistent), then for all skeletons $\mathcal{M}'$, W is $(\mathcal{M} \otimes \mathcal{M}')$-prefix-independent (resp. $(\mathcal{M} \otimes \mathcal{M}')$-cycle-consistent).*

Moreover, an ω-regular language recognized by a parity automaton $(\mathcal{M}, p)$ is $\mathcal{M}$-prefix-independent and $\mathcal{M}$-cycle-consistent.

Main results. We state our main technical tool. We recall that *one-player $\mathcal{M}$-determinacy* of a winning condition W is both about one-player arenas of $\mathcal{P}_1$ (trying to achieve a word in W) *and* of $\mathcal{P}_2$ (trying to achieve a word in $\overline{W}$).

▶ **Theorem 8.** *Let $W \subseteq C^\omega$ be a winning condition and $\sim$ be its right congruence.*

1. *If there exists a skeleton $\mathcal{M}$ such that W is one-player $\mathcal{M}$-determined, then $\sim$ has finite index (in particular, W is $\mathcal{M}_\sim$-prefix-independent) and W is $\mathcal{M}$-cycle-consistent.*
2. *If there exists a skeleton $\mathcal{M}$ such that W is $\mathcal{M}$-prefix-independent and $\mathcal{M}$-cycle-consistent, then W is ω-regular and can be recognized by a deterministic parity automaton defined on top of $\mathcal{M}$.*

Technical sketch. We prove the first and second items of this theorem in [9, Sections 4 and 5]. We comment briefly on our proof technique for each item.

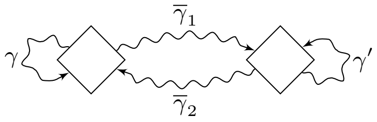


Figure 2 Comparing cycles γ and γ' using intermediate cycle $\overline{\gamma} = \overline{\gamma}_1\overline{\gamma}_2$. Squiggly arrows indicate a sequence of transitions. Cycles γ and $\gamma\overline{\gamma}_1\overline{\gamma}_2$ are winning, and cycles γ' and $\gamma'\overline{\gamma}_2\overline{\gamma}_1$ are losing.

1. For the first item, we assume that W is one-player $\mathcal{M}$-determined for a skeleton $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$. We define a preorder $\preceq$ on C^* such that $w_1 \preceq w_2$ if $w_1^{-1}W \subseteq w_2^{-1}W$. Notice that the right congruence $\sim$ of W is equal to $\preceq \cap \succeq$. By exhibiting well-chosen one-player arenas, using the $\mathcal{M}$-determinacy assumption, we can show that for each $m \in M$, in the set $\mathsf{c}\Pi_{m_{\mathsf{init}},m}$, relation $\preceq$ is total and there is no infinite increasing nor decreasing sequence (for $\preceq$). This shows that $\sim$ has finite index on each $\mathsf{c}\Pi_{m_{\mathsf{init}},m}$; as M is finite and $C^* = \bigcup_{m\in M} \mathsf{c}\Pi_{m_{\mathsf{init}},m}$, relation $\sim$ has finite index on C^*. The proof of $\mathcal{M}$-cycle-consistency is more direct: if a player had an interest in mixing multiple losing cycles of $\mathcal{M}$ to make them into a winning play, we could find a (possibly infinite) one-player arena of that player in which strategies based on $\mathcal{M}$ would not suffice to play optimally.

2. For the second item, we assume that W is $\mathcal{M}$-prefix-independent and $\mathcal{M}$-cycle-consistent for a skeleton $\mathcal{M}$. Our technical lemmas focus on *cycles* of $\mathcal{M}$, how they relate to each other, and what happens when we combine them. Our main tool is to define a partial preorder on cycles, which will help assign priorities to transitions of $\mathcal{M}$ – the aim being to define a parity condition on top of $\mathcal{M}$ that recognizes W. As we consider $\mathcal{M}$-prefix-independence along with $\mathcal{M}$-cycle-consistency, for m a state of $\mathcal{M}$, each cycle in Γ_m has a well-defined accepting status: it generates either a winning or a losing infinite word when repeated infinitely often after any finite word in $\mathsf{c}\Pi_{m_{\mathsf{init}},m}$.

Intuitively, for some state m of $\mathcal{M}$, for γ a winning cycle on m and γ' a losing cycle on m, we can look at which cycle *dominates* the other, that is, whether the combined cycle $\gamma\gamma'$ is winning, in which case γ dominates γ', or losing, in which case γ' dominates γ ($\gamma\gamma'$ and $\gamma'\gamma$ necessarily have the same accepting status). This shows how to compare cycles with different accepting statuses that start on the same skeleton state. This notion and some properties about this notion generalize part of the proof technique of [17], in which colors rather than cycles are compared.

We can extend this idea to some pairs of a winning cycle γ and a losing cycle γ' that have no state in common: our criterion to compare two such cycles is that there is a cycle $\overline{\gamma}$ connecting them such that $\overline{\gamma}$ is not “powerful enough” to alter the values of each cycle separately, that is, such that $\gamma\overline{\gamma}$ is winning and $\gamma'\overline{\gamma}$ is losing. To know which cycle dominates the other, we look at the accepting value of the cycle $\gamma\overline{\gamma}_1\gamma'\overline{\gamma}_2$, for some adequate break of $\overline{\gamma}$ into two paths $\overline{\gamma}_1$ and $\overline{\gamma}_2$. We illustrate the situation in Figure 2. If $\gamma\overline{\gamma}_1\gamma'\overline{\gamma}_2$ is winning, then γ dominates γ', and if it is losing, then γ' dominates γ.

This defines a partial preorder on cycles of $\mathcal{M}$. We show that there is no infinite decreasing nor increasing sequence for this preorder, and after defining a related equivalence relation, that there are finitely many equivalence classes of cycles. We can assign finitely many priorities to these cycles in a way consistent with the partial preorder, and then transfer these priorities to *transitions* of $\mathcal{M}$, as a function $p\colon M \times C \to \{0, \ldots, n\}$. We conclude by showing that W is recognized by parity automaton $(\mathcal{M}, p)$. ◀

We state two consequences of Theorem 8: a strategic characterization of ω-regular languages, and a novel one-to-two-player-lift.

▶ **Theorem 9** (Characterization). *Let $W \subseteq C^\omega$ be a language of infinite words. Language W is ω-regular if and only if it is chromatic-finite-memory determined (in infinite arenas).*

Proof. One implication is well-known [36, 42]: if W is ω-regular, then it can be recognized by a deterministic parity automaton whose skeleton we can use as a memory that suffices to play optimally for both players, in arenas of any cardinality. The other direction is given by Theorem 8: if W is chromatic-finite-memory determined, then there exists in particular a skeleton $\mathcal{M}$ such that W is one-player $\mathcal{M}$-determined, so $\sim$ has finite index and W is $\mathcal{M}$-cycle-consistent. In particular, by Lemma 7, W is $(\mathcal{M}_\sim \otimes \mathcal{M})$-prefix-independent and $(\mathcal{M}_\sim \otimes \mathcal{M})$-cycle-consistent, so W is ω-regular and can be recognized by a deterministic parity automaton defined on top of $\mathcal{M}_\sim \otimes \mathcal{M}$. ◀

▶ **Theorem 10** (One-to-two-player lift). *Let $W \subseteq C^\omega$ be a winning condition. Language W is **one-player** chromatic-finite-memory determined if and only if it is chromatic-finite-memory determined.*

Proof. The implication from two-player to one-player arenas is trivial. The other implication is given by Theorem 8: if W is one-player $\mathcal{M}$-determined, then $\sim$ has finite index and W is $\mathcal{M}$-cycle-consistent. Again by Lemma 7 and Theorem 8, as W can be recognized by a parity automaton defined on top of $\mathcal{M}_\sim \otimes \mathcal{M}$, W is determined and strategies based on $\mathcal{M}_\sim \otimes \mathcal{M}$ suffice to play optimally in all two-player arenas. ◀

We discuss two specific situations in which we can easily derive interesting consequences using our results: the prefix-independent case, and the case where the minimal-state automaton suffices to play optimally.

Prefix-independent case. If a condition W is prefix-independent (i.e., $\sim$ has index 1 and $\mathcal{M}_\sim = \mathcal{M}_{\mathsf{triv}}$), and skeleton $\mathcal{M}$ suffices to play optimally in one-player games, then W is recognized by a parity automaton defined on top of $\mathcal{M}_{\mathsf{triv}} \otimes \mathcal{M}$, which is isomorphic to $\mathcal{M}$. This implies that the exact same memory can be used by both players to play optimally in two-player arenas, with no increase in memory. Note that we do not know in general whether this product is necessary to go from one-player to two-player arenas, but the question is automatically solved for prefix-independent conditions.

If, moreover, $\mathcal{M} = \mathcal{M}_{\mathsf{triv}}$ (i.e., memoryless strategies suffice to play optimally in one-player arenas), we recover exactly the result from Colcombet and Niwiński [17]: W can be recognized by a parity automaton defined on top of $\mathcal{M}_{\mathsf{triv}}$, so we can directly assign a priority to each color with a function $p\colon C \to \{0, \dots, n\}$ such that an infinite word $w = c_1 c_2 \ldots \in C^\omega$ is in W if and only if $\limsup_{i \geq 1} p(c_i)$ is even.

$\mathcal{M}_\sim$-determined case. An interesting property of some ω-regular languages is that they can be recognized by defining an acceptance condition on top of the minimal-state automaton of their right congruence [35], which is a useful property for the learning of languages [1]. Here, Theorem 8 shows that W can be recognized by defining a transition-based parity acceptance condition on top of the minimal-state automaton $\mathcal{M}_\sim$ if and only if W is $\mathcal{M}_\sim$-determined. The transition-based parity acceptance condition was not considered in the cited results [35, 1].

▶ **Corollary 11.** *Let $W \subseteq C^\omega$ be an ω-regular language and $\mathcal{M}_\sim$ be the minimal-state automaton of its right congruence. The following are equivalent:*

1. *W is recognized by defining a transition-based parity acceptance condition on top of $\mathcal{M}_\sim$;*
2. *W is $\mathcal{M}_\sim$-determined;*
3. *W is $\mathcal{M}_\sim$-cycle-consistent.*

Proof. Implication 1. $\implies$ 2. follows from the memoryless determinacy of parity games [42]. Implication 2. $\implies$ 3. follows from the first item of Theorem 8. Implication 3. $\implies$ 1. follows from the second item of Theorem 8: by definition, W is $\mathcal{M}_\sim$-prefix-independent; if it is also $\mathcal{M}_\sim$-cycle-consistent, then W is recognized by a parity automaton defined on top of $\mathcal{M}_\sim$. ◀

Classes of arenas. We discuss the sensitivity of Theorem 8 w.r.t. our model of arenas.

There are multiple conditions that are chromatic-finite-memory determined if we only consider *finite arenas* (finitely many states and edges) and not infinite arenas. A few examples are discounted-sum games [40], mean-payoff games [19], total-payoff games [22], one-counter games [10] which are all memoryless-determined in finite arenas but which require infinite memory to play optimally in some infinite arenas (we discuss some of these in Section 4). In particular, Theorem 9 tells us that the derived winning conditions are not ω-regular.

Strangely, the fact that our arenas have colors on *edges* and not on *states* is crucial for the result. Indeed, there exists a winning condition (a generalization of a parity condition with infinitely many priorities [25]) that is memoryless-determined in state-labeled infinite arenas, but not in edge-labeled infinite arenas (as we consider here). This particularity was already discussed [17], and it was also shown that the same condition is memoryless-determined in edge-labeled arenas with finite branching. Therefore, the fact that we allow *infinite branching* in our arenas is also necessary for Theorem 9. Another example of a winning condition with finite memory requirements in finitely branching arenas for one player but infinite memory requirements in infinitely branching arenas is presented in [16, Section 4].

4 Applications

We provide applications of our results to discounted-sum and mean-payoff conditions.

4.1 Discounted sum

We apply our results to a *discounted-sum* condition in order to illustrate our notions. A specificity of this example is that its ω-regularity depends on some parameters – we use our results to characterize the parameters for which it is ω-regular or, equivalently (Theorem 9), chromatic-finite-memory determined. The ω-regularity of discounted-sum conditions has also been studied in [14, 2] with different techniques and goals.

Let $C \subseteq \mathbb{Q}$ be non-empty and bounded. For $\lambda \in (0, 1) \cap \mathbb{Q}$, we define the *discounted-sum function* $\mathsf{DS}_\lambda \colon C^\omega \to \mathbb{R}$ such that for $w = c_1c_2 \ldots \in C^\omega$, $\mathsf{DS}_\lambda(w) = \sum_{i=1}^{\infty} \lambda^{i-1} \cdot c_i$. This function is always well-defined for a bounded C, and takes values in $[\frac{\inf C}{1-\lambda}, \frac{\sup C}{1-\lambda}]$.

We define the winning condition $\mathsf{DS}_\lambda^{\geq 0} = \{w \in C^\omega \mid \mathsf{DS}_\lambda(w) \geq 0\}$ as the set of infinite words whose discounted sum is non-negative, and let $\sim$ be its right congruence. We will analyze cycle-consistency and prefix-independence of $\mathsf{DS}_\lambda^{\geq 0}$ to conclude under which conditions (on C and λ) it is chromatic-finite-memory determined (or equivalently, ω-regular by Theorem 9). First, we discuss a few properties of the discounted-sum function.

Basic properties. We extend function DS_λ to finite words in a natural way: for $w \in C^*$, we define $\mathsf{DS}_\lambda(w) = \mathsf{DS}_\lambda(w0^\omega)$. For $w \in C^*$, we define $|w|$ as the length of w (so $w \in C^{|w|}$). First, we notice that for $w \in C^*$ and $w' \in C^\omega$, we have $\mathsf{DS}_\lambda(ww') = \mathsf{DS}_\lambda(w) + \lambda^{|w|}\mathsf{DS}_\lambda(w')$. Therefore, $ww' \in \mathsf{DS}_\lambda^{\geq 0}$ if and only if $\frac{\mathsf{DS}_\lambda(w)}{\lambda^{|w|}} \geq -\mathsf{DS}_\lambda(w')$. This provides a characterization of the winning continuations of a finite word $w \in C^*$ by comparing their discounted sum to the value $\frac{\mathsf{DS}_\lambda(w)}{\lambda^{|w|}}$.

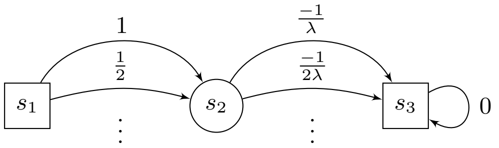


Figure 3 Arena with infinitely many edges in which $\mathcal{P}_1$ needs infinite memory to win for condition $\mathsf{DS}_\lambda^{\geq 0}$ from s_1 for any $\lambda \in (0,1) \cap \mathbb{Q}$, with $C = [-k,k] \cap \mathbb{Q}$ for k sufficiently large.

This leads us to define the *gap* of a finite word $w \in C^*$, following ideas in [5], as

$$\mathsf{gap}(w) = \begin{cases} \top & \text{if } \frac{\mathsf{DS}_\lambda(w)}{\lambda^{|w|}} \geq -\frac{\inf C}{1-\lambda}, \\ \bot & \text{if } \frac{\mathsf{DS}_\lambda(w)}{\lambda^{|w|}} < -\frac{\sup C}{1-\lambda}, \\ \frac{\mathsf{DS}_\lambda(w)}{\lambda^{|w|}} & \text{otherwise.} \end{cases}$$

Intuitively, the gap of a finite word $w \in C^*$ represents how far it is from going back to 0: if $w' \in C^\omega$ is such that $\mathsf{DS}_\lambda(w') = -\mathsf{gap}(w)$, then $\mathsf{DS}_\lambda(ww') = 0$. We can see that for all words $w \in C^*$, if $\mathsf{gap}(w) = \top$, then all continuations are winning (i.e., $w^{-1}W = C^\omega$) as it is not possible to find an infinite word with a discounted sum less than $\frac{\inf C}{1-\lambda}$. Similarly, if $\mathsf{gap}(w) = \bot$, then all continuations are losing (i.e., $w^{-1}W = \emptyset$).

Cycle-consistency. We have that $\mathsf{DS}_\lambda^{\geq 0}$ is $\mathcal{M}_{\mathsf{triv}}$-cycle-consistent (proof in [9, Section 6]).

▶ **Proposition 12.** *For all bounded $C \subseteq \mathbb{Q}$, $\lambda \in (0,1) \cap \mathbb{Q}$, winning condition $\mathsf{DS}_\lambda^{\geq 0}$ is $\mathcal{M}_{\mathsf{triv}}$-cycle-consistent.*

Prefix-independence. If $C = [-k,k] \cap \mathbb{Q}$ for some $k \in \mathbb{N} \setminus \{0\}$, winning condition $\mathsf{DS}_\lambda^{\geq 0}$ is not $\mathcal{M}$-prefix-independent for any $\mathcal{M}$, as $\sim$ has infinite index. Indeed, we have for instance that elements in $\{\frac{1}{i} \in C^* \mid i \geq 1\}$ are all in different equivalence classes of $\sim$. We can see how to use this to exhibit an arena in which $\mathcal{P}_1$ can win but needs infinite memory to do so in Figure 3.

For finite $C \subseteq \mathbb{Z}$, the picture is more complicated; for $C = [-k,k] \cap \mathbb{Z}$ for some $k \in \mathbb{N}$, we characterize when $\mathsf{DS}_\lambda^{\geq 0}$ is $\mathcal{M}$-prefix-independent for some finite skeleton $\mathcal{M}$. We give an intuition of the two situations in which that happens: (i) if C is too small, then the first non-zero color seen determines the outcome of the game, as it is not possible to compensate this color to change the sign of the discounted sum; (ii) if $\lambda = \frac{1}{n}$ for some integer $n \geq 1$, then the gap function actually takes only finitely many values, which is not the case for a different λ.

▶ **Proposition 13.** *Let $\lambda \in (0,1) \cap \mathbb{Q}$, $k \in \mathbb{N}$, and $C = [-k,k] \cap \mathbb{Z}$. Then, the right congruence $\sim$ of $\mathsf{DS}_\lambda^{\geq 0}$ has finite index if and only if $k < \frac{1}{\lambda} - 1$ or λ is equal to $\frac{1}{n}$ for some integer $n \geq 1$.*

Proof (sketch). Full proof in [9, Section 6]. The key property is to show that gaps characterize equivalence classes of prefixes: for $w_1, w_2 \in C^*$, $w_1 \sim w_2$ if and only if $\mathsf{gap}(w_1) = \mathsf{gap}(w_2)$. Once this is proven, it is left to determine the number of different gap values in each situation, which corresponds to the index of $\sim$. We illustrate one situation in which the index is finite by depicting the minimal-state automaton of $\sim$ for $\lambda = \frac{1}{2}$ and $k = 2 \geq \frac{1}{\lambda} - 1$ in Figure 4. ◀

Connecting Propositions 12 and 13, here is the characterization we obtain using Theorem 8.

▶ **Corollary 14.** *Let $\lambda \in (0,1) \cap \mathbb{Q}$, $k \in \mathbb{N}$, and $C = [-k,k] \cap \mathbb{Z}$. Condition $\mathsf{DS}_\lambda^{\geq 0}$ is chromatic-finite-memory determined (or equivalently, ω-regular) if and only if $k < \frac{1}{\lambda} - 1$ or λ is equal to $\frac{1}{n}$ for some integer $n \geq 1$.*

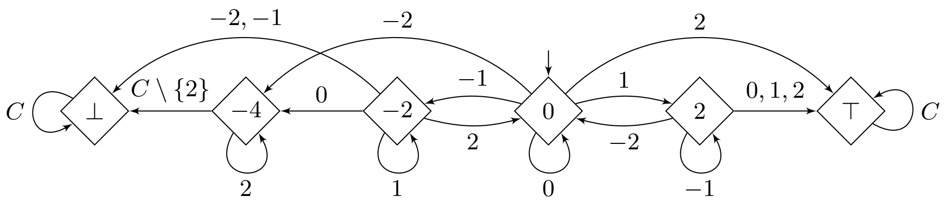


Figure 4 Minimal-state automaton of $\sim$ for $\lambda = \frac{1}{2}$ and $C = \{-2, -1, 0, 1, 2\}$. The value in a state is the gap value characterizing the equivalence class of $\sim$. Here, $\frac{\sup C}{1-\lambda} = 4$ and $\frac{\inf C}{1-\lambda} = -4$. The asymmetry around 0 comes from the ≥ 0 in the definition of the condition: when state -4 is reached, there is exactly one winning continuation (2^ω), but a state with gap value 4 would only have winning continuations (hence, it is part of state $\top$). Notice that we can define a parity condition on top of this automaton that recognizes $\mathsf{DS}_\lambda^{\geq 0}$: an infinite word is winning as long as it does not reach $\bot$.

4.2 Mean payoff

Let $C \subseteq \mathbb{Q}$ be non-empty. We define the *mean-payoff function* $\mathsf{MP}\colon C^\omega \to \mathbb{R} \cup \{-\infty, \infty\}$ such that for $w = c_1 c_2 \ldots \in C^\omega$, $\mathsf{MP}(w) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} c_i$. We define the winning condition $\mathsf{MP}^{\geq 0} = \{w \in C^\omega \mid \mathsf{MP}(w) \geq 0\}$ as the set of infinite words whose mean payoff is non-negative. This condition is $\mathcal{M}_{\mathsf{triv}}$-prefix-independent for any set of colors. However, it is known that infinite-memory strategies may be required to play optimally in some infinite arenas [39, Section 8.10]; the example provided uses infinitely many colors. Here, we show that chromatic-finite-memory strategies do not suffice to play optimally, even for $C = \{-1, 1\}$. Let us analyze cycle-consistency of $\mathsf{MP}^{\geq 0}$. If we consider, for $n \in \mathbb{N}$,

$$w_n = \underbrace{1 \ldots 1}_{n \text{ times}} \underbrace{-1 \ldots -1}_{n+1 \text{ times}},$$

we have that $(w_n)^\omega$ is losing for all $n \in \mathbb{N}$, but the infinite word $w_0 w_1 w_2 \ldots$ has a mean payoff of 0 and is thus winning. This shows directly that $\mathsf{MP}^{\geq 0}$ is not $\mathcal{M}_{\mathsf{triv}}$-cycle-consistent. The argument can be adapted to show that $\mathsf{MP}^{\geq 0}$ is not $\mathcal{M}$-cycle-consistent for any skeleton $\mathcal{M}$ (see [9, Section 6]).

5 Conclusion

We proved an equivalence between chromatic-finite-memory determinacy of a winning condition in games on infinite graphs and ω-regularity of the corresponding language of infinite words, generalizing a result by Colcombet and Niwiński [17]. A "strategic" consequence is that chromatic-finite-memory determinacy in one-player games of both players implies the seemingly stronger chromatic-finite-memory determinacy in zero-sum games. A "language-theoretic" consequence is a link between the representation of ω-regular languages by parity automata and the memory structures used to play optimally in zero-sum games, using as a tool the minimal-state automata classifying the equivalence classes of the right congruence.

For future work, one possible improvement over our result is to deduce tighter chromatic memory requirements in two-player games compared to one-player games. Our proof technique gives as an upper bound on the two-player memory requirements a product between the minimal-state automaton and a sufficient skeleton for one-player arenas, but smaller skeletons often suffice. We do not know whether the product with the minimal-state automaton is necessary in general in order to play optimally in two-player arenas (although it is necessary in Theorem 8 to describe W using a parity automaton). This behavior contrasts with the case of finite arenas, in which it is known that a skeleton sufficient for both players in finite one-player arenas also suffices in finite two-player arenas [6, 8]. More generally, it would be interesting to characterize precisely the (chromatic) memory requirements of ω-regular winning conditions, extending work on the subclass of Muller conditions [18, 11].

References

1 Dana Angluin and Dana Fisman. Regular ω-languages with an informative right congruence. *Inf. Comput.*, 278:104598, 2021. doi:10.1016/j.ic.2020.104598.

2 Suguman Bansal, Swarat Chaudhuri, and Moshe Y. Vardi. Comparator automata in quantitative verification. *CoRR*, abs/1812.06569, 2018. arXiv:1812.06569.

3 Alessandro Bianco, Marco Faella, Fabio Mogavero, and Aniello Murano. Exploring the boundary of half-positionality. *Ann. Math. Artif. Intell.*, 62(1-2):55–77, 2011. doi:10.1007/s10472-011-9250-1.

4 Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. Graph games and reactive synthesis. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 921–962. Springer, 2018. doi:10.1007/978-3-319-10575-8_27.

5 Udi Boker, Thomas A. Henzinger, and Jan Otop. The target discounted-sum problem. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 750–761. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.74.

6 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 24:1–24:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.24.

7 Patricia Bouyer, Stéphane Le Roux, and Nathan Thomasset. Finite-memory strategies in two-player infinite games. *CoRR*, abs/2107.09945, 2021. arXiv:2107.09945.

8 Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 26:1–26:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.26.

9 Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. *CoRR*, abs/2110.01276, 2021. arXiv:2110.01276.

10 Tomás Brázdil, Václav Brozek, and Kousha Etessami. One-counter stochastic games. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 108–119. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.108.

11 Antonio Casares. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. *CoRR*, abs/2105.12009, 2021. arXiv:2105.12009.

12 Antonio Casares, Thomas Colcombet, and Nathanaël Fijalkow. Optimal transformations of games and automata using Muller conditions. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 123:1–123:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.123.

13 Krishnendu Chatterjee and Laurent Doyen. Perfect-information stochastic games with generalized mean-payoff objectives. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 247–256. ACM, 2016. doi:10.1145/2933575.2934513.

14 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 199–208. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.16.

15 Krishnendu Chatterjee and Nathanaël Fijalkow. Infinite-state games with finitary conditions. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 181–196. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.181.

16 Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 379–390. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.379.

17 Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theor. Comput. Sci.*, 352(1-3):190–196, 2006. doi:10.1016/j.tcs.2005.10.046.

18 Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

19 Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *Int. Journal of Game Theory*, 8(2):109–113, 1979. doi:10.1007/BF01768705.

20 Hugo Gimbert. Parity and exploration games on infinite graphs. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 56–70. Springer, 2004. doi:10.1007/978-3-540-30124-0_8.

21 Hugo Gimbert and Edon Kelmendi. Submixing and shift-invariant stochastic games. *CoRR*, abs/1401.6575, 2014. arXiv:1401.6575.

22 Hugo Gimbert and Wieslaw Zielonka. When can you play positionally? In Jirí Fiala, Václav Koubek, and Jan Kratochvíl, editors, *Mathematical Foundations of Computer Science 2004, 29th International Symposium, MFCS 2004, Prague, Czech Republic, August 22-27, 2004, Proceedings*, volume 3153 of *Lecture Notes in Computer Science*, pages 686–697. Springer, 2004. doi:10.1007/978-3-540-28629-5_53.

23 Hugo Gimbert and Wieslaw Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005. doi:10.1007/11539452_33.

24 Hugo Gimbert and Wieslaw Zielonka. Pure and stationary optimal strategies in perfect-information stochastic games with global preferences. Unpublished, 2009. URL: https://hal.archives-ouvertes.fr/hal-00438359.

25 Erich Grädel and Igor Walukiewicz. Positional determinacy of games with infinitely many priorities. *Log. Methods Comput. Sci.*, 2(4), 2006. doi:10.2168/LMCS-2(4:6)2006.

26 Stefan Kiefer, Richard Mayr, Mahsa Shirmohammadi, Patrick Totzke, and Dominik Wojtczak. How to play in infinite MDPs (invited talk). In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 3:1–3:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.3.

27 Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure Appl. Log.*, 69(2-3):243–268, 1994. doi:10.1016/0168-0072(94)90086-8.

28 Eryk Kopczyński. Half-positional determinacy of infinite games. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2006. doi:10.1007/11787006_29.

29 Eryk Kopczyński. Omega-regular half-positional winning conditions. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2007. doi:10.1007/978-3-540-74915-8_7.

30 Eryk Kopczyński. *Half-positional Determinacy of Infinite Games.* PhD thesis, Warsaw University, 2008.

31 Alexander Kozachinskiy. One-to-two-player lifting for mildly growing memory. *CoRR*, abs/2104.13888, 2021. arXiv:2104.13888.

32 Stéphane Le Roux. Time-aware uniformization of winning strategies. In Marcella Anselmo, Gianluca Della Vedova, Florin Manea, and Arno Pauly, editors, *Beyond the Horizon of Computability - 16th Conference on Computability in Europe, CiE 2020, Fisciano, Italy, June 29 - July 3, 2020, Proceedings*, volume 12098 of *Lecture Notes in Computer Science*, pages 193–204. Springer, 2020. doi:10.1007/978-3-030-51466-2_17.

33 Stéphane Le Roux and Arno Pauly. Extending finite-memory determinacy to multi-player games. *Inf. Comput.*, 261(Part):676–694, 2018. doi:10.1016/j.ic.2018.02.024.

34 Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.38.

35 Oded Maler and Ludwig Staiger. On syntactic congruences for omega-languages. *Theor. Comput. Sci.*, 183(1):93–112, 1997. doi:10.1016/S0304-3975(96)00312-X.

36 Andrzej Wlodzimierz Mostowski. Regular expressions for infinite trees and a standard form of automata. In Andrzej Skowron, editor, *Computation Theory - Fifth Symposium, Zaborów, Poland, December 3-8, 1984, Proceedings*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1984. doi:10.1007/3-540-16066-3_15.

37 A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. doi:10.2307/2033204.

38 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.32.

39 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley Series in Probability and Statistics. Wiley, 1994. doi:10.1002/9780470316887.

40 L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953. doi:10.1073/pnas.39.10.1095.

41 Ludwig Staiger. Finite-state omega-languages. *J. Comput. Syst. Sci.*, 27(3):434–448, 1983. doi:10.1016/0022-0000(83)90051-X.

42 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.

On Testing Decision Tree

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

Catherine A. Haddad-Zaknoon
Department of Computer Science, Technion, Haifa, Israel

Abstract

In this paper, we study testing decision tree of size and depth that are significantly smaller than the number of attributes n.

Our main result addresses the problem of poly$(n, 1/\epsilon)$ time algorithms with poly$(s, 1/\epsilon)$ query complexity (independent of n) that distinguish between functions that are decision trees of size s from functions that are ϵ-far from any decision tree of size $\phi(s, 1/\epsilon)$, for some function $\phi > s$. The best known result is the recent one that follows from Blanc, Lange and Tan, [3], that gives $\phi(s, 1/\epsilon) = 2^{O((\log^3 s)/\epsilon^3)}$. In this paper, we give a new algorithm that achieves $\phi(s, 1/\epsilon) = 2^{O(\log^2(s/\epsilon))}$.

Moreover, we study the testability of depth-d decision tree and give a *distribution free* tester that distinguishes between depth-d decision tree and functions that are ϵ-far from depth-d^2 decision tree.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees

Keywords and phrases Testing decision trees

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.17

1 Introduction

Decision tree is one of the popular predictive modelling approaches used in many areas including statistics, data mining and machine learning. Recently, property-testing of subclasses of decision trees have attracted much attention [1, 3, 6, 9, 10, 12]. In property testing, the algorithm is provided by an access to a black box to some Boolean function f and labeled random examples of f according to some distribution $\mathcal{D}$. Given a subclass of decision trees C, we need to decide whether f is a decision tree in C or "far" from being in C with respect to $\mathcal{D}$, [4, 11, 13].

Since finding efficient algorithms for this problem is difficult, the following relaxation is considered. Let H be a larger class of decision trees $H \supset C$. Then, we are interested in the question: can we efficiently test C by H? That is, to efficiently decide whether f is a decision tree in C or "far" from being in H with respect to $\mathcal{D}$, [12]. In this context, the challenge is to find a small class $H \supset C$ such that efficient testing algorithm exists. In this paper, we address this problem while examining and constructing algorithms that are efficient in the query complexity and run in polynomial time.

1.1 Models

Let C and $H \supseteq C$ be two classes of Boolean functions $f : \{0,1\}^n \to \{0,1\}$. In the *distribution-free* model, the algorithm has an access to a *black box query* and *random example query*. The black box query, for an input $x \in \{0,1\}^n$ returns $f(x)$. The random example query, when invoked, returns a random example $(x, f(x))$ such that x is chosen according to an arbitrary and unknown distribution $\mathcal{D}$. In the *uniform distribution* model, $\mathcal{D} = U$ is the uniform distribution over $\{0,1\}^n$.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 17; pp. 17:1–17:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A function $g \in H$ is called ϵ-close to $f \in C$ with respect to the distribution $\mathcal{D}$ if, $\mathbf{Pr}_{\mathcal{D}}[g(x) \neq f(x)] \leq \epsilon$. In the *distribution-free property testing C by H*, [12], (resp. uniform distribution property testing), for *any* Boolean function f, we need to distinguish, with high probability and via the above queries to f, between the case that f is in C versus the case that f is ϵ-far (not ϵ-close) from every function in H with respect to $\mathcal{D}$ (resp. the uniform distribution). Such an algorithm is called a *tester for C by H*. When $H = C$, the tester is called a *tester for C*.

1.2 Decision Tree

A *decision tree* is a rooted binary tree in which each internal node is labeled with a variable x_i and has two children. Each leaf is labeled with an output from $\{0, 1\}$. A decision tree computes a Boolean function in the following way: given an input $x \in \{0, 1\}^n$, the value of the function on x is the output in the leaf reached by the path that starts at the root and goes left or right at each internal node according to whether the variable's value in x is 0 or 1, respectively.

The *size* of a decision tree is the number of leaves of the tree. The *depth* of a node (resp. leaf) in a decision tree is the number of edges in the path from the root to the node (resp. leaf). The depth of the tree is the maximum over all the depth values of its leaves. A *depth-d decision tree T* is a decision tree of depth at most d. A *size-s decision tree T* is a decision tree of size at most s.

1.3 Other Representations of Decision Tree

A *monomial* is a conjunction of variables, and a *term* is a conjunction of literals (variable and negated variable). Two terms t_1 and t_2 are called *disjoint* if $t_1 \wedge t_2 = 0$ (over the field F_2). A *multilinear polynomial* (or just a polynomial) is Boolean function that is defined as a sum of monomials over the field F_2. Every Boolean function can be expressed uniquely as a multilinear polynomial. A *disjoint-terms sum* is a sum of disjoint terms. Every Boolean function can be represented as a disjoint-terms sum. The representation is not unique.

A decision tree f can be represented as a disjoint-terms sum according to the following recurrence. If the decision tree is a leaf, then its disjoint-terms sum representation is the constant in this leaf. If the root label of f is x_i then, $f = x_i f_1 + \overline{x_i} f_0$ where f_1 and f_0 are the disjoint-terms sum of right and left sub-trees of f respectively. It is easy to see that the number of terms we get in this recurrence is equal to the number of leaves in the tree labeled with 1. Therefore, every leaf corresponds to a term and the number of literals in this term is equal to the depth of the leaf. To represent a disjoint-terms sum as a polynomial, we write for each appearance of $\overline{x_i}$ as $x_i + 1$ and expand the expressions with the regular arithmetic rules in the field F_2.

2 Main Result and Technique

In this section, we present previous and new results for testing decision trees. In the next two subsections, we consider two significant results in testing.

2.1 Testing Decision Tree of Size s

Let f be a Boolean function over the variables $x_1, \ldots, x_n$. For $x_{i_1}, \ldots, x_{i_j}$ and $\xi_1, \ldots, \xi_j \in \{0, 1\}$, denote by $f_{|x_{i_1} \leftarrow \xi_1, \ldots, x_{i_j} \leftarrow \xi_j}$ the function that results from substituting $x_{i_r} = \xi_r$, $r = 1, \ldots, j$ in f.

In [3], Blanc, Lange and Tan give the first tester that runs in $\mathrm{poly}(n, 1/\epsilon)$ time with $\mathrm{poly}(\log s,\ 1/\epsilon)\log n$ query complexity and distinguishes between functions that are s-size decision tree and functions that are ϵ-far from every size-$\phi(s, 1/\epsilon)$ decision tree for some function $\phi(s, 1/\epsilon)$. When $n \gg s$, one can use the reduction from [6] to get a $\mathrm{poly}(n, 1/\epsilon)$ time algorithm with $\mathrm{poly}(s, 1/\epsilon)$ query complexity (independent of n) for the same problem. The function ϕ achieved in [3] is $\phi(s, 1/\epsilon) = 2^{(\log^3 s)/\epsilon^3}$. We use a different approach to get $\phi(s, 1/\epsilon) = 2^{\log^2(s/\epsilon)}$.

In fact, Blanc, Lange and Tan [3] solve a more challenging problem. Their tester is a tolerant tester. That is, it distinguishes between functions that are ϵ-close to s-size decision tree and functions that are $\Omega(\epsilon)$-far from every size-$\phi(s, 1/\epsilon)$ decision tree.

To achieve their result, Blanc et. al. define for every function f a complete decision tree $T(d, f)$ of depth $d = O(\log^3 s/\epsilon^3)$ as follows. They define the noise sensitivity of f, $\mathrm{NS}(f)$, as the probability that $f(x) \neq f(x + y)$ where x is uniform random, and for every i, $y_i = 1$ with probability $p = O(\epsilon/\log s)$. The score of f with respect to x_i, $\mathrm{Score}_i(f)$, is defined to be the expected decrease in the noise sensitivity of f provided that x_i is queried. The label of the root of $T(d, f)$ is selected to be the variable x_i that maximizes the score. The left and right sub-trees of $T(d, f)$ are $T(d-1, f_{|x_i\leftarrow 0})$ and $T(d-1, f_{|x_i\leftarrow 1})$, respectively. Then $T(0, g)$ is defined to be a leaf labeled with 0 if $\mathbf{E}[g] < 1/2$ and 1 otherwise. Here g is the function that results from f by substituting the partial assignment defined by the path from the root to the leaf. They prove that, for $d = O(\log^3 s/\epsilon^3)$, if f is a size-s decision tree, then f is $\epsilon/4$-close to $T(d, f)$, and if f is ϵ-far from every size-$2^{O(\log^3 s/\epsilon^3)}$ then, since $T(d, f)$ is size-2^d $\left(= 2^{O(\log^3 s/\epsilon^3)}\right)$ decision tree, f is ϵ-far from $T(d, f)$. Moreover, they show that a query to $T(d, f)$ can be done in $\mathrm{poly}(n, 1/\epsilon)$ time and $\mathrm{poly}(\log s, 1/\epsilon)\log n$ queries. Therefore, $T(d, f)$ and f can be queried to test if they are $\epsilon/4$-close or ϵ-far. By applying the reduction from [6], a tester that solves the same problem in $\mathrm{poly}(n, 1/\epsilon)$ time and $\mathrm{poly}(s, 1/\epsilon)$ queries to f is obtained.

In this paper, we use a different approach. Let f be a size-s decision tree. Our algorithm regards f as a polynomial. Although f may have exponential number of monomials, we are interested in the influential ones only, that is, the small monomials. The number of small monomials in the polynomial representation of size-s decision tree may be exponential. To control the number of small monomials, we shuffle the monomials by choosing a uniform random $a \in \{0, 1\}^n$ and considering $T(x) = f(x + a)$. In disjoint-terms sum representation of f (see Subsection 1.3), a large term t in f (a term that results from a leaf of depth $\Omega(\log(s/\epsilon))$ in the tree), with high probability (w.h.p), more than quarter of its variables become positive in T, and therefore, it only generates large monomials. Therefore, the shuffling process ensures that the number of significant monomials in T is as small as $\mathrm{poly}(s/\epsilon)$. This technique is used in [8] for learning decision tree under the uniform distribution.

Let c be a large constant and let F be the sum of the monomials of size $r = c\log(s/\epsilon)$ in $T(x)$. Let G be the sum of monomials of size greater than r and less than $16r$ in $T(x)$. First, we run the algorithm of Bshouty and Mansour in [8] to exactly learn F and G in polynomial time. If the learning algorithm fails, then w.h.p, f is not size-s decision tree and the algorithm rejects. Then, we define a decision tree $T(d, F, G)$ of depth $d = O(\log^2(s/\epsilon))$ as follows. Define $\mathrm{Frac}(F, x_i)$ to be the fraction of the number of monomials in F that contain x_i. Choose a variable x_{i_1} with the minimum index i_1 that maximizes $\mathrm{Frac}(F, x_{i_1})$ and use it as the label of the root of $T(d, F, G)$. The left and right sub-trees of $T(d, F, G)$ are $T(d-1, F^{(0)}, G^{(0)})$ and $T(d-1, F^{(1)}, G^{(1)})$, respectively, where $F^{(\xi)}$ is the sum of all monomials that appear in $F_{|x_{i_1}\leftarrow\xi}$ and not in $G_{|x_{i_1}\leftarrow\xi}$, and $G^{(\xi)}$ is the sum of all monomials that appear in $G_{|x_{i_1}\leftarrow\xi}$ and not in $F_{|x_{i_1}\leftarrow\xi}$.

We are interested in making a random walk in this tree. Assume we are given the first $m-1$ steps in the random walk $\xi^{(m-1)} = (\xi_1, \xi_2, \dots, \xi_{m-1}) \in \{0,1\}^{m-1}$, $F^{\xi^{(m-1)}}$ and $G^{\xi^{(m-1)}}$. We find the variable with the minimum index i_m that maximizes $\mathrm{Frac}(F^{\xi^{(m-1)}}, x_{i_m})$. Choose a random uniform $\xi_m \in \{0,1\}$. Then, for $\xi^{(m)} = (\xi_1, \xi_2, \dots, \xi_m)$, $F^{\xi^{(m)}}$ is defined to be the sum of all the monomials in $F^{\xi^{(m-1)}}_{|x_{i_m} \leftarrow \xi_m}$ that are not in $G^{\xi^{(m-1)}}_{|x_{i_m} \leftarrow \xi_m}$, and $G^{\xi^{(m)}}$ is defined to be the sum of all the monomials in $G^{\xi^{(m-1)}}_{|x_{i_m} \leftarrow \xi_m}$ that are not in $F^{\xi^{(m-1)}}_{|x_{i_m} \leftarrow \xi_m}$. If $F^{\xi^{(m)}}$ is a constant function $\eta \in \{0,1\}$, then we have reached a leaf labeled with η. The way we define $F^{\xi^{(m)}}$ and $G^{\xi^{(m)}}$ turns to be crucial in the algorithm and is needed for its correctness proof. We note here that, unlike the tester of Blanc, Lange and Tan, this tree is not the decision tree of T or F, because every path in the tree treats F and G as sets of monomials rather than functions.

We show that, when f is a size-s decision tree, for a random shuffling and random $\xi = (\xi_1, \xi_2, \dots, \xi_m) \in \{0,1\}^m$, with high probability, $\mathrm{Frac}(F^{\xi^{(m)}}, x_{i_m})$ is at least $1/O(\log(s/\epsilon))$. Hence, for a random walk in the tree $T(d, F, G)$, each step decreases the number of monomials in $F^{\xi^{(m)}}$ by a factor of $1 - 1/O(\log(s/\epsilon))$ on average. Therefore, since F contains at most $poly(s/\epsilon)$ monomials, with high probability, a random walk in $T(d, F, G)$ reaches a leaf in $O(\log^2(s/\epsilon))$ steps.

Now suppose f is ϵ-far from every size-$2^{O(\log^2(s/\epsilon))}$ decision tree. It might happen that a function f that is ϵ-far from size-$2^{O(\log^2(s/\epsilon))}$ decision tree passes all the above tests, because the above algorithm relies only on the small monomials of T. Moreover, it might happen that the small monomials of such function coincide with the monomials of a small size decision tree. As a result, we add another test at each leaf of the tree that checks if the function T at the leaf of the tree $T_{|x_{i_1} \leftarrow \xi_1, \dots, x_{i_m} \leftarrow \xi_m}$ is $\epsilon/4$-close to a constant function.

For a function that is ϵ-far from every size-$2^{O(\log^2(s/\epsilon))}$ decision tree, if it is not rejected because its small monomials coincide with the monomial of small size decision tree, then the random walks will often reach a small depth leaf. On the other hand, if almost all the small depth leaves give a good approximation of the function, then T is ϵ-close to a small depth tree. Therefore, the function is rejected with high probability.

We also show that, although the tester treats F and G as sets of monomials and not as functions, if f is size-s decision tree then the tree gives a good approximation of T.

The above tester runs in $poly(n, 1/\epsilon)$ time and queries. Using the reduction in [6], it can be changed to a tester that runs in $poly(s, 1/\epsilon)n$ time and makes $poly(s, 1/\epsilon)$ queries.

2.2 Testing Decision Tree of Depth d

The algorithm of Blanc, Lange and Tan [3] also distinguishes between depth-d decision tree and functions that are ϵ-far from depth-$O(d^3/\epsilon^3)$ decision trees under the uniform distribution. Again, we can make the query complexity independent of n using the reduction of Bshouty, [6], and get a $2^{O(d)}n$ time uniform-distribution tester that asks $2^{O(d)}/\epsilon$ queries and distinguishes between depth-d decision tree and functions that are ϵ-far from depth-$O(d^3/\epsilon^3)$ decision trees.

In this paper, we give a new simple *distribution-free* tester that runs in $2^{O(d)}n$ time, asks $2^{O(d)}/\epsilon$ queries and distinguishes between depth-d decision tree and functions that are ϵ-far from depth-d^2 decision trees.

Our algorithm relies on the following fact. Let f be a depth-d decision tree. Consider the polynomial representation of $f = M_1 + M_2 + \cdots + M_m$. For any maximal monomial $M_i = x_{i_1} x_{i_2} \cdots x_{i_t}$ (a monomial that is not sub-monomial of any other monomial in f) and any $\xi_1, \dots, \xi_t \in \{0,1\}$, the function $f_{|x_{i_1} \leftarrow \xi_1, \dots, x_{i_t} \leftarrow \xi_t}$ is depth $(d-1)$-decision tree.

We can define a depth-d^2 decision tree T_f that is equivalent to f as follows: Find a maximal monomial $M_i = x_{i_1}x_{i_2}\cdots x_{i_t}$. Then use all its variables in the first t levels of the decision tree T_f. That is, build a complete tree of depth t that all its nodes at level j are labeled with x_{i_j}. This defines a different path for each $x_{i_1} = \xi_1, \ldots, x_{i_t} = \xi_t$. Then, in the last node of such path, attach the tree T_g for $g = f_{|x_{i_1}\leftarrow\xi_1,\ldots,x_{i_t}\leftarrow\xi_t}$. Since depth-$d$ decision trees are degree-d polynomials, we have $t \le d$, and since the decision trees at level t are depth-$(d-1)$ decision trees, the depth of T_f is at most d^2 (in fact it is at most $d(d-1)/2$).

For the tester, we will not construct T_f, but instead, we show that for any assignment a, finding the route that a takes in the tree T_f can be done efficiently. For this end, we first show that if f is a depth-d decision tree then, the relevant variables of f can be found in $\tilde{O}(2^{2d}) + 2^d\log n$ queries. Then, we show that a maximal monomial of f can be found in $\tilde{O}(2^d)$ queries. For an assignment a drawn according to a distribution $\mathcal{D}$, if f is a depth-d decision tree, the route that a takes in T_f ends before depth d^2. If f is ϵ-far from depth-d decision tree, then, either finding the relevant variables of f fails, or finding a maximal monomial of size at most d fails or, with probability at least ϵ, the route in T_f goes beyond depth d^2. The later happens because if it does not for $O(1/\epsilon)$ examples drawn according to a distribution $\mathcal{D}$, then truncating the tree up to depth d^2, results a tree that is w.h.p ϵ-close of a depth d^2-decision tree with respect to $\mathcal{D}$.

Notice that the query complexity of this tester depends on n because finding the relevant variables of f takes $\tilde{O}(2^{2d}) + 2^d\log n$ queries which depends on n. To make the query complexity independent of n, we use the reduction of Bshouty in [6].

2.3 Non-Polynomial Time Testers

A recent breakthrough result of Blanc et. al. [2] with the reduction of Bshouty [6] gives a uniform tester for size-s decision tree that runs in $n(s/\epsilon)^{O(\log((\log s)/\epsilon))}$ time and makes $(s/\epsilon)^{O(\log((\log s)/\epsilon))}$ queries.

3 A Tester for Depth-d Decision Tree

In this section we prove the following result:

▶ **Theorem 1.** *There is a distribution-free tester that makes $q = \tilde{O}(2^{2d}/\epsilon)$ queries to unknown function f, runs in $O(qn)$ time and*

1. *Accepts w.h.p if f is a depth-d decision tree.*
2. *Rejects w.h.p if f is ϵ-far from depth-d^2 decision trees.*

3.1 The Key Lemma

We start with some notations and definitions, and then prove the key Lemma for the tester.

Recall that *monomial* is a conjunction of variables. A k-monomial is a monomial with at most k variables. A polynomial (over the field F_2) is a sum (in the binary field F_2) of monomials. An s-sparse polynomial is a sum of at most s monomials. We say that the polynomial f is of degree-d if its monomials are d-monomials.

We say that x_i is *relevant* variable in f if $f_{|x_i\leftarrow 0} \ne f_{|x_i\leftarrow 1}$. It is well known that (see for example Lemma 4 in [7]):

▶ **Lemma 2.** *A depth-d decision tree is a 3^d-sparse degree-d polynomial with at most 2^d relevant variables.*

Let $f = M_1 + M_2 + \cdots + M_t$ be a polynomial. We say that M_i is a *maximal monomial* of f if M_i is not a sub-monomial of any other monomial M_j, i.e., for every other monomial M_j, there is a variable in M_i that is not in M_j.

We now prove the key Lemma for our tester:

▶ **Lemma 3.** *Let f be a depth-d decision tree and $f = M_1 + M_2 + \cdots + M_t$ be its polynomial representation. Let $M_i = x_{i_1} \cdots x_{i_{d'}}$, $d' \le d$ be a maximal monomial of f. For any $\xi_1, \xi_2, \ldots, \xi_{d'} \in \{0, 1\}$, we have that $f_{|x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}}$ is depth-$(d-1)$ decision tree.*

Proof. The proof is by induction on the number of variables m of f. The case $m = 1$ is trivial. Now assume that the result holds for any $m \le k$.

Let T be any depth-d decision tree with $k+1$ variables that represents f. Let $M_i = x_{i_1} \cdots x_{i_{d'}}$, $d' \le d$ be a maximal monomial of f. Let $X = \{x_{i_1}, \ldots, x_{i_{d'}}\}$. If the variable of the root of T is $x_{i_j} \in X$ then, $T_{|x_{i_j} \leftarrow 0}$ and $T_{|x_{i_j} \leftarrow 1}$ are left and right decision sub-trees of T and are of depth at most $d-1$. Then, $T_{|x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}}$ is of depth at most $d-1$ for any $\xi_1, \ldots, \xi_{d'} \in \{0, 1\}$.

If the variable of the root of the tree T is $x_\ell \notin X$, then the left sub-tree $T_{|x_\ell \leftarrow 0}$ is a depth-$(d-1)$ decision tree and has at most k variables. We now claim that M_i is a maximal monomial of $T_{|x_\ell \leftarrow 0}$. This is because of the fact that substituting $x_\ell = 0$ in the polynomial representation only removes monomials in f. Since x_ℓ is not in M_i, it does not remove M_i. Therefore, M_i is maximal monomial in $T_{|x_\ell \leftarrow 0}$, and by the induction hypothesis $T_{|x_\ell \leftarrow 0, x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}}$ is depth-$(d-2)$ decision tree.

The right sub-tree $T_{|x_\ell \leftarrow 1}$ is a depth-$(d-1)$ decision tree that has at most k variables. We now claim that $T_{|x_\ell \leftarrow 1}$ also has M_i as a monomial and it is maximal. Assume for the sake of contradiction that M_i is removed or not maximal, then there must be a monomial $M_j = x_\ell M_i$ in T. Since M_i is sub-monomial of M_j, we get a contradiction to the fact that M_i is maximal in f. Therefore, by the induction hypothesis $T_{|x_\ell \leftarrow 1, x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}}$ is depth-$(d-2)$ decision tree. This implies that

$$T_{|x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}} = x_\ell \cdot T_{|x_\ell \leftarrow 1, x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}} + \overline{x_\ell} \cdot T_{|x_\ell \leftarrow 0, x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}}$$

is depth-$(d-1)$ decision tree. ◀

For every degree-d polynomial f, we define the following decision tree T_f. If f is constant function, then T_f is a leaf labeled with this constant. Let $f = M_1 + M_2 + \cdots + M_t$. Consider any maximal monomial M_i of f. Let $M_i = x_{i_1} \cdots x_{i_{d'}}$ where $i_1 < i_2 < \cdots < i_{d'}$. The tree T_f has all the variables $x_{i_1}, \cdots, x_{i_{d'}}$ at the first d' levels of the tree. That is, the first d' levels of the tree is a complete tree where the label of all the nodes at level j is x_{i_j}. So every $\xi_1, \ldots, \xi_{d'} \in \{0, 1\}$ leads to a different vertex at level d' in T_f from which we recursively attach the decision tree T_g where $g = f_{|x_{i_1} \leftarrow \xi_1, \cdots, x_{i_{d'}} \leftarrow \xi_{d'}}$.

We now prove:

▶ **Lemma 4.** *Let f be a degree-d polynomial. Then, for $h = d(d-1)/2$:*

1. *If f is a depth-d decision tree, then T_f is depth h-decision tree.*
2. *If f is ϵ-far from every depth $h+1$ decision tree according to a distribution $\mathcal{D}$ then, for a random assignment a drawn according to the distribution $\mathcal{D}$, with probability at least ϵ, the path that a takes in T_f reaches depth $h+1$.*

Proof. **1.** follows immediately from Lemma 3.

To prove **2.**, let T'_f be the tree T_f where every vertex of depth $h+1$ is changed to a leaf labeled with 0. Since f is ϵ-far from every depth $h+1$ decision tree according to the distribution $\mathcal{D}$, it is ϵ far from T'_f. Since the leaves of T'_f of depth at most h correctly compute f, the probability that a random assignment a chosen according to distribution $\mathcal{D}$ ends up in a leaf of depth at most h is less than $1-\epsilon$. This completes the proof. ◀

3.2 The Tester

In this section, we prove Theorem 1. To that end, we start by the following Lemma:

▶ **Lemma 5.** *We have*

1. *There is an algorithm that for a degree-d polynomial f makes $q=\tilde{O}(2^{2d})+2^d\log n$ queries, runs in time $O(qn)$ and finds the relevant variables of f.*
2. *There is an algorithm that for a degree-d polynomial over 2^d variables X makes $q'=\tilde{O}(2^d)$ queries, runs in time $O(q'n)$ and finds a maximal monomial in f.*

The proof of Lemma 5 is given in subsection 3.3. This immediately gives the following result that we need for our tester.

▶ **Lemma 6.** *Let f be a sparse-3^d degree-d polynomial over 2^d variables. Let $h=d(d-1)/2$. Given the relevant variables of f, for any $a\in\{0,1\}^n$, the path that a takes in T_f up to depth at most $h+1$ can be computed in $\tilde{O}(2^d)h$ time.*

The tester's paradigm is as follows. First, the tester finds the relevant variables of f. If the number of relevant variables exceeds 2^d, then the tester rejects. The tester then, for $t=O(1/\epsilon)$ assignments $a^{(1)},\dots,a^{(t)}$ drawn according to the distribution $\mathcal{D}$, finds the route of each $a^{(i)}$ in T_f. If no maximal monomial of size at most d can be found then the tester rejects. If one of the routes exceeds depth $h=d(d-1)/2$, the algorithm rejects. Otherwise it accepts. Each route takes time $\tilde{O}(2^d)$. So the number of queries is $q=\tilde{O}(2^{2d})/\epsilon+2^d\log n$ and the time is $O(qn)$. We now use the reduction of Bshouty in [6] to make the query complexity independent of n and get the result. See Lemma 26 in Appendix A.

3.3 Proof of Lemma 5

In this subsection, we prove Lemma 5. We show how to find the relevant variables and a maximal monomial of any degree-d polynomial.

The following is a very well known result [8]:

▶ **Lemma 7.** *For any non-constant degree-d polynomial f over F_2, we have $\boldsymbol{Pr}[f(x)\neq f(0)]\geq 1/2^d$.*

The following is a well known result in learning theory. We prove it for completeness.

▶ **Lemma 8.** *There is an algorithm that given any degree-d polynomial f over v variables and a set X of some of its relevant variables, asks $2^d\log(1/\delta)+\log v$ queries and, with probability at least $1-\delta$, decides if the variables in X are all its relevant variables, and if not, finds a new relevant variable of f.*

Proof. Let $X'=\{x_{i_1},\dots,x_{i_t}\}$ be the set of variables that are not in X. Define $g=f+f_{|X'\leftarrow 0}$. Since g is of degree at most d, by Lemma 7, with $2^d\log(1/\delta)$ queries to g, with probability at least $1-\delta$, we can decide if g is a constant function. If not, we get an assignment a such that $g(a)\neq g(0)$. If g is constant function $\tau\in\{0,1\}$, then $f(x)=f_{|X'\leftarrow 0}+\tau$ and f is independent of X'. So, the variables in X are all the relevant variables of f

If $g(a) \neq g(0)$, then since $g(0) = 0$, we have $f(a) \neq f_{|X' \leftarrow 0}(a) = f(a_{|X' \leftarrow 0})$. Now recursively flip half of the entries of a that differ from $a_{|X' \leftarrow 0}$ and ask a query and keep the two assignments that have different values in f. Eventually, we get an entry $a_{i_{k+1}}$ that flipping it changes the value of the function. Then, $x_{i_{k+1}}$ is relevant variable in f. Now $x_{i_{k+1}} \notin X$ is a new relevant variable because the entries of each $x_{i_j} \in X$ in a agree with the value of the same entry in $a_{|X' \leftarrow 0}$. The number of queries in this procedure is at most $\log v$. This completes the proof. ◀

Therefore, for degree-d polynomial, choosing confidence $\delta/2^d$ in the above algorithm, with probability at least $1 - \delta$, we find the first 2^d relevant variables of f using $\tilde{O}(2^{2d}) \log(1/\delta) + 2^d \log n$ queries. If f has more than 2^d relevant variables, then f is not a depth-d decision tree and the tester rejects.

We now prove

▶ **Lemma 9.** *Let f be a degree-d polynomial and X be the set of its relevant variables. Let $M = x_{i_1} \cdots x_{i_k}$ be a sub-monomial of some monomial of f (M is not necessarily a monomial of f). There is an algorithm that asks $O(2^d \log(1/\delta) + 2^k \log |X|)$ queries and, with probability at least $1 - \delta$, decides if M is a maximal monomial of f, and if it is not, it finds a new variable $x_{i_{k+1}}$ such that $M' = x_{i_1} \cdots x_{i_k} x_{i_{k+1}}$ is a sub-monomial of some monomial of f.*

Proof. Define the function $G(x) = 1 + \sum_{(\xi_1,\ldots,\xi_k)\in\{0,1\}^k} f_{|x_{i_1}\leftarrow\xi_1,\cdots,x_{i_k}\leftarrow\xi_k}(x)$. We prove the following:

1. A query to G can be simulated by 2^k queries to f.
2. G is a polynomial of degree at most $d - k$.
3. M is maximal monomial of f if and only if $G = 0$.
4. If $G \neq 0$, then for any relevant variable $x_{i_{k+1}}$ of G, $M' = x_{i_1} \cdots x_{i_k} x_{i_{k+1}}$ is a sub-monomial of some monomial of f.

The first item is obvious. We prove 2-4. Since M is a sub-monomial of some monomial of f, we have $f = Mg + h$, where g is a polynomial of degree at most $d - k$ (independent of $x_{i_1}, \ldots, x_{i_k}$) and h is a polynomial of degree at most d that M is not sub-monomial of any of its monomials. Notice that M is maximal monomial of f if and only if $g = 1$. For a monomial M'' in h, we have that some variable in M, say w.l.o.g. x_{i_1}, is not in M'' and therefore,

$$\sum_{(\xi_1,\ldots,\xi_k)\in\{0,1\}^k} M''_{|x_{i_1}\leftarrow\xi_1,\cdots,x_{i_k}\leftarrow\xi_k}(x) = \sum_{\xi_1\in\{0,1\}} \sum_{(\xi_2,\ldots,\xi_k)\in\{0,1\}^{k-1}} M''_{|x_{i_2}\leftarrow\xi_2,\cdots,x_{i_k}\leftarrow\xi_k}(x) = 0.$$

Denote by $\xi := (\xi_1, \cdots, \xi_k) \in \{0,1\}^k$, then we can write:

$$\begin{aligned} G(x) + 1 &= \sum_{\xi\in\{0,1\}^k} f_{|x_{i_1}\leftarrow\xi_1,\cdots,x_{i_k}\leftarrow\xi_k}(x) \\ &= g(x) \sum_{\xi\in\{0,1\}^k} M_{|x_{i_1}\leftarrow\xi_1,\cdots,x_{i_k}\leftarrow\xi_k}(x) + \sum_{\xi\in\{0,1\}^k} h_{|x_{i_1}\leftarrow\xi_1,\cdots,x_{i_k}\leftarrow\xi_k}(x) \\ &= g(x) + \sum_{(M'' \text{ monomial in } h)} \sum_{(\xi_1,\ldots,\xi_k)\in\{0,1\}^k} M''_{|x_{i_1}\leftarrow\xi_1,\cdots,x_{i_k}\leftarrow\xi_k}(x) \\ &= g(x). \end{aligned}$$

Hence,$f = MG + M + h$ and the results 2-4 follows.

By Lemma 8, there is an algorithm that asks $2^{d-k}\log(1/\delta) + \log |X|$ queries to G (and therefore, $O(2^d \log(1/\delta) + 2^k \log |X|)$ queries to f) and, with probability at least $1 - \delta$, either decides that $G = 0$, in which case M is maximal monomial, or finds a new relevant variable $x_{i_{k+1}}$ of G, in which case $M' = x_{i_1} \cdots x_{i_k} x_{i_{k+1}}$ is a sub-monomial of some monomial of f. ◀

For degree-d polynomials with $|X| \le 2^d$ relevant variables, we choose confidence $\delta/2^d$ in the above algorithm and then, with probability at least $1-\delta$, we find a maximal monomial of f.

4 A Tester for Size-s Decision Tree

In this section we prove:

▶ **Theorem 10.** *There is a (uniform-distribution) tester that makes $q = poly(s, 1/\epsilon)$ queries to unknown function f, runs in $poly(s, 1/\epsilon)n$ time and*

1. *Accepts w.h.p if f is a size-s decision tree.*
2. *Rejects w.h.p if f is ϵ-far from size-$(s/\epsilon)^{O(\log(s/\epsilon))}$ decision trees.*

4.1 Preliminary Results

For a Boolean function f, the *constant-depth of f*, $cd(f)$, is the minimum number ℓ of variables $X = \{x_{j_1}, \cdots, x_{j_\ell}\}$ such that $f_{|X\leftarrow 0}$ is a constant function. We define $\mathcal{M}(f)$ the set of all monomials in the minimum size polynomial representation of f. Since minimum size polynomial representation of a Boolean function is unique, $\mathcal{M}(f)$ is well defined. For a set of monomials S, we denote $\Sigma S = \sum_{M\in S} M$. Notice that $\Sigma\mathcal{M}(f) = f$ and $\mathcal{M}(\Sigma S) = S$. For any Boolean function and an interval I (such as $[d] = \{1, 2, \ldots, d\}$, $[d_1, d_2] = [d_2]\backslash[d_1 - 1]$ or $(d_1, d_2] = [d_2]\backslash[d_1]$), we define $f^I = \sum_{M\in\mathcal{M}(f) \text{ and } |M|\in I} M$ where $|M|$ is the number of variables in M. We first prove:

▶ **Lemma 11.** *For any $S' \subseteq \mathcal{M}(f)$, we have $cd(\Sigma S') \le cd(f)$. In particular, for any interval I, we have $cd(f^I) \le cd(f)$.*

Proof. Let $cd(f) = r$. Then, there is a set $X = \{x_{j_1}, \cdots, x_{j_r}\}$ such that $f_{|X\leftarrow 0}$ is constant. Therefore, for every non-constant $M \in \mathcal{M}(f)$, we have $M_{|X\leftarrow 0} = 0$. Then, $(\Sigma S')_{|X\leftarrow 0}$ is constant and $cd(\Sigma S') \le cd(f)$. ◀

▶ **Lemma 12.** *Let f be any Boolean function. If $cd(f) \le \ell$, then there is a variable x_i that appears in at least $1/\ell$ fraction of the non-constant monomials of f.*

Proof. If $cd(f) \le \ell$, then there is a set $X = \{x_{j_1}, \cdots, x_{j_\ell}\}$ such that $f_{|X\leftarrow 0}$ is a constant function. This implies that for every non-constant monomial, there is $x_i \in X$ that appears in it. By the pigeonhole principle the result follows. ◀

Let $a \in \{0,1\}^n$ be a random uniform assignment. Consider, $T(x) = f(x + a)$. Then:

▶ **Lemma 13.** *Let f be a size-s decision tree and let $T(x) = f(x + a)$ for a random uniform assignment $a \in \{0,1\}^n$. For a random uniform $\xi_1, \ldots, \xi_j \in \{0,1\}$ and any variables $x_{i_1}, \ldots, x_{i_j}$ where each i_ℓ may depend on T, a, $i_1, \ldots, i_{\ell-1}$ and $\xi_1, \ldots, \xi_{\ell-1}$ but is independent of $\xi_\ell, \ldots, \xi_j$ and $q = (x_{i_1} \leftarrow \xi_1, \ldots, x_{i_j} \leftarrow \xi_j)$, with probability at least $1 - s2^{-h}$, $cd\left(T_{|q}\right) \le h$.*

Proof. We have $T_{|x_{i_1}\leftarrow\xi_1,\ldots,x_{i_j}\leftarrow\xi_j}(0) = T(0_{|x_{i_1}\leftarrow\xi_1,\ldots,x_{i_j}\leftarrow\xi_j}) = f(a + 0_{|x_{i_1}\leftarrow\xi_1,\ldots,x_{i_j}\leftarrow\xi_j})$. Since $b := a + 0_{|x_{i_1}\leftarrow\xi_1,\ldots,x_{i_j}\leftarrow\xi_j}$ is random uniform in $\{0,1\}^n$, the path that b takes in the computation of $f(b)$ is a random uniform path in f. With probability at least $1 - s2^{-h}$, this path reaches a leaf at depth less than or equal to h in f. Therefore, with probability at least $1 - s2^{-h}$, there are $h' \le h$ variables $x_{j_1}, \ldots, x_{j_{h'}}$ (the variables in this path) such that $f_{x_{j_1}\leftarrow b_{j_1},\ldots,x_{j_{h'}}\leftarrow b_{j_{h'}}}$ is constant, say $\tau \in \{0,1\}$. Let $J = \{j_1, j_2, \ldots, j_{h'}\}$ and $I = \{i_1, i_2, \ldots, i_j\}$ and suppose, w.l.o.g, $J \cap I = \{i_1, \ldots, i_r\}$. Then,

$$\begin{aligned}\tau &= f_{|x_{j_1}\leftarrow b_{j_1},\ldots,x_{j_{h'}}\leftarrow b_{j_{h'}}}(x) = f_{|x_{j_1}\leftarrow b_{j_1},\ldots,x_{j_{h'}}\leftarrow b_{j_{h'}}}(x+a)\\ &= T_{|x_{j_1}\leftarrow b_{j_1}+a_{j_1},\ldots,x_{j_{h'}}\leftarrow b_{j_{h'}}+a_{j_{h'}}} = \left(T_{|x_{i_1}\leftarrow \xi_1,\ldots,x_{i_r}\leftarrow \xi_r}\right)_{|(J\backslash I)\leftarrow 0}\end{aligned}$$

and thus,

$$\begin{aligned}(T_{|q})_{|(J\backslash I)\leftarrow 0} &= (T_{|x_{i_1}\leftarrow \xi_1,\ldots,x_{i_j}\leftarrow \xi_j})_{|(J\backslash I)\leftarrow 0}\\ &= (T_{|x_{i_1}\leftarrow \xi_1,\ldots,x_{i_r}\leftarrow \xi_r})_{|(J\backslash I)\leftarrow 0, x_{i_{r+1}}\leftarrow \xi_{r+1},\ldots,x_{i_j}\leftarrow \xi_j} = \tau.\end{aligned}$$

Therefore, $cd\left(T_{|q}\right) \le |J\backslash I| \le h' \le h$. ◀

Let

$$r = \log(s/\epsilon). \tag{1}$$

The tester uses Lemma 13, $poly(\log(s/\epsilon))/\epsilon$ times, therefore we can choose $h = 2r$ which, by union bound, adds a failure probability of $(poly(\log(s/\epsilon))/\epsilon)\cdot s2^{-h} = \tilde{O}(\epsilon/s)$ to the tester. Let E_1 be the event

$$E_1:\quad (\forall q)\ cd(T_{|q}) \le 2r, \tag{2}$$

for all the q that are generated in the tester.

For the rest of this section, we let f be a size-s decision tree. Let $f = f_1 + f_2 + \cdots + f_s$ be the disjoint-terms sum representation of f.[1] Let $T_i = f_i(x+a)$ for $i \in [s]$. It is easy to see that $T = T_1 + T_2 + \cdots + T_s$ is disjoint-terms sum representation of T. We denote by T_i^+ the conjunction of the non-negated variables in T_i. We prove:

▶ **Lemma 14.** *Let λ be any constant. For a random uniform a, with probability at least $1 - s(\epsilon/s)^\lambda$ the following event $E_2(\lambda)$ holds*

$$E_2(\lambda):\quad \textit{For every } i\textit{, if } |T_i| > 16\lambda r \textit{ then } |T_i^+| \ge 4\lambda r. \tag{3}$$

Proof. Since $T_i(x) = f_i(x+a)$ and a is random uniform, each variable in T_i is positive with probability 1/2. By Chernoff bound the result follows. ◀

To change the disjoint-terms representation of T to polynomial representation, we take every term T_i and expand it to sum of monomials[2]. A monomial that is generated from even number of different terms will not appear in the polynomial, while, those that are generated from odd number of different terms will appear in the polynomial.

Let $M_1 + M_2 + \cdots + M_\ell$ be the multivariate polynomial representation of T, where $|M_1| \le |M_2| \le \cdots \le |M_\ell|$. Note that ℓ can be exponential in n. We say that M_i *is generated by* T_j if j is the smallest integer for which T_j generates M_i. The following is a trivial result:

▶ **Lemma 15.** *If M_i is generated by T_j, then $|T_j| \ge |M_i| \ge |T_j^+|$ and T_j^+ is a sub-monomial of M_i. That is, $M_i = T_j^+ M_i'$ for some monomial M_i'.*

[1] Every size-s decision tree can be represented as a sum of terms $T_1 + T_2 + \cdots + T_{s'}$, $s' \le s$, where $T_i \wedge T_j = 0$ for every $i \ne j$. The number of terms s' is the number of leaves labeled with 1. See for example [8]. Here we assume $s' = s$ because we can always change a term t to $tx_j + t\overline{x_j}$.

[2] For example $x_1x_2\bar{x}_3\bar{x}_4 = x_1x_2(x_3+1)(x_4+1) = x_1x_2x_3x_4 + x_1x_2x_3 + x_1x_2x_4 + x_1x_2$.

▶ **Lemma 16.** *If $E_2(\lambda)$ in (3) holds, then the number of monomials M_i of size at most $4\lambda r$ is at most $s(s/\epsilon)^{16\lambda}$.*

Proof. By Lemma 14, the monomials that has size at most $4\lambda r$ are generated from terms of size at most $16\lambda r$. We have at most s terms of size at most $16\lambda r$ and each one generates at most $2^{16\lambda r}$ monomials. So we have at most $s(s/\epsilon)^{16\lambda}$ such monomials. ◀

▶ **Lemma 17.** *If $E_2(\lambda)$ in (3) holds, then there are s monomials $N_1, N_2, \ldots, N_s$, each of size at least $4\lambda r$, such that for every monomial M_i (of T) of size at least $16\lambda r$, there is N_j where $M_i = N_j M_i'$ for some monomial M_i'.*

Proof. Let $N_i = T_i^+$ if $|T_i^+| \geq 4\lambda rn$ and $N_i = x_1 x_2 \cdots x_n$ otherwise. Let M_i be a monomial of T of size at least $16\lambda r$. By Lemma 15, M_i is generated by a monomial T_j of size at least $16\lambda r$. By Lemma 14, $|T_j^+| \geq 4\lambda r$. By Lemma 15, $N_j = T_j^+$ is a sub-monomial of M_i. ◀

We remind the reader that for an interval R, T^R is the sum of the monomials M of T of size $|M| \in R$. For a set of monomials A, we denote $\vee A = \vee_{M\in A} M$. We also need the following lemma:

▶ **Lemma 18.** *Suppose $E_2(\lambda)$ holds. Let $P = \vee\mathcal{M}(T^{(16\lambda r, s]})$. For a random uniform $\xi_1, \ldots, \xi_j \in \{0,1\}$ and any variables $x_{i_1}, \ldots, x_{i_j}$ where each i_ℓ may depend on T, $i_1, \ldots, i_{\ell-1}$ and $\xi_1, \ldots, \xi_{\ell-1}$ but independent of $\xi_\ell, \ldots, \xi_j$ and $q = (x_{i_1} \leftarrow \xi_1, \ldots, x_{i_j} \leftarrow \xi_j)$, with probability at least $1 - s(\epsilon/s)^{2\lambda}$, $\boldsymbol{Pr}\left[P_{|q} = 1\right] \leq s\left(\frac{\epsilon}{s}\right)^{2\lambda}$.*

Proof. By Lemma 17, we have $P = N_1 P_1 \vee N_2 P_2 \vee \cdots \vee N_s P_s$, where P_i is a Boolean function and N_i is a monomial of size at least $4\lambda r$ for each $i \in \{1, \ldots, s\}$. The probability that each $(N_i)_{|q}$ is not zero and is of size at most $2\lambda r$ is at most $2^{-2\lambda r} = (\epsilon/s)^{2\lambda}$. Since, for all $i \in [s]$, $(N_i)_{|q}$ is zero or $|(N_i)_{|q}| > 2\lambda r$ implies $\mathbf{Pr}[P_{|q} = 1] \leq s(\epsilon/s)^{2\lambda}$, the result follows. ◀

4.2 The Tester

In this subsection, we give the tester and prove its correctness. Recall that $r = \log(s/\epsilon)$. Let $c \geq 2$ be any constant. The tester first chooses a random uniform $a \in \{0,1\}^n$ and defines $T(x) = f(x + a)$. Then, it learns all the monomials of size $16r'$ where[3] $r' = 16cr$. This can be done by the algorithm in [8] in $poly(n, s/\epsilon)$ time and queries. We show later how to eliminate n in the query complexity. Then, the tester splits the monomials of size at most $16r'$ to monomials of size less or equal to r' (the function F) and those that have size between r' and $16r'$ (the function G). The tester performs $O(1/\epsilon)$ random walks in a decision tree. For each stage j, the set H_j is defined in a way that (1) it is a subset of the monomials of the function $F_{|x_{i_1}\leftarrow\xi_1,\ldots,x_{i_{j-1}}\leftarrow\xi_{j-1}}$ and (2) it contains a variable that appears in at least $1/(2r)$ fraction of the monomials. At each stage j, the tester deterministically chooses a variable x_{i_j} with the smallest index i_j that appears in at least $1/(2r)$ fraction of the monomials of ΣH_j and chooses a random $\xi_j \in \{0,1\}$ for x_{i_j}. See the details in Algorithm 1.

Although this decision tree is not the decision tree of F, we can still show that when f is size-s decision tree, with probability at least $2/3$ the random walk ends after $O(\log^2(s/\epsilon))$ steps and then the tester accepts. When it is ϵ-far from any size-$(s/\epsilon)^{O(\log(s/\epsilon))}$ decision tree then, with probability at least $2/3$, something goes wrong (the learning algorithm fails or no variable appears in at least $1/(2r)$ fraction of the monomials of H_j) or the random walk does not end after $\Omega(\log^2(s/\epsilon))$ steps and then it rejects.

[3] Here c can be 2. We kept it to show the effect of this constant on the success probability of the tester.

Algorithm 1 : Test(f) - A tester for size-s decision tree.

```
Input: Black box access to $f$
Output: Accept or Reject
$T \leftarrow f(x+a)$ for random uniform $a \in \{0,1\}^n$.
Learn $T^{[16r']}$. If FAIL then Reject.
$F \leftarrow T^{[r']}$; $G \leftarrow T^{(r',16r']}$; $H_0 \leftarrow \mathcal{M}(F)$; $L_0 \leftarrow \mathcal{M}(G)$.
for $i = 1$ to $40/\epsilon$ do
    $j \leftarrow 0$; $q_0 \leftarrow$ Empty sequence.
    while $\Sigma H_j$ is not constant and $j < 2^{10}c(\log^2(s/\epsilon))$ do
        $j \leftarrow j+1$.
        Find a variable $x_{i_j}$ with the smallest index that appears in at least $1/(2r)$ fractions of the monomials in $H_{j-1}$.
        If no such variable exists then Reject.
        Choose a random uniform $\xi_j \in \{0,1\}$.
        $q_j \leftarrow (q_{j-1}; x_{i_j} \leftarrow \xi_j)$.
                    ▷ I.e., add to the list of substitutions $q_{j-1}$ the substitution $x_{i_j} \leftarrow \xi_j$.
        $H_j \leftarrow \mathcal{M}((\Sigma H_{j-1})_{|x_{i_j}\leftarrow\xi_j}) \backslash \mathcal{M}((\Sigma L_{j-1})_{|x_{i_j}\leftarrow\xi_j})$
        $L_j \leftarrow \mathcal{M}((\Sigma L_{j-1})_{|x_{i_j}\leftarrow\xi_j}) \backslash \mathcal{M}((\Sigma H_{j-1})_{|x_{i_j}\leftarrow\xi_j})$
    end while
    if $j = 2^{10}c(\log^2(s/\epsilon))$ then
        Reject.
    end if
end for
if  $\mathbf{Pr}[T_{|q_j} = 1]$ is in $[\epsilon/4, 1-\epsilon/4]$ then
    Reject
end if
Accept.
```

The tester query complexity is $poly(n, s/\epsilon)$. We use the reduction from [6] to change the query complexity to $poly(s/\epsilon)$.

▶ **Lemma 19.** *Assume that the event $E_2(16c)$ in (3) holds. Let $F = T^{[r']}$ and $G = T^{(r',16r']}$. Let $x_{i_1}, \ldots, x_{i_j}$ be variables, $\xi_1, \ldots, \xi_j$ be random uniform values in $\{0,1\}$, $q_j = (x_{i_1} \leftarrow \xi_1, \ldots, x_{i_j} \leftarrow \xi_j)$, $q_{j+1} = (q_j, x_{i_{j+1}} \leftarrow \xi_{j+1})$, H_j, H_{j+1} and L_j as defined in the procedure **Test**(f) in Algorithm 1. Then, with probability at least $1 - s(\epsilon/s)^{3c}$, we have*

1.

$$H_j \subseteq \mathcal{M}\left((T_{|q_j})^{[r']}\right). \tag{4}$$

Let E be the event that (4) holds for all $j \leq 2^{10}c\log^2(s/\epsilon)$ and all the $40/\epsilon$ random walks of the tester. Then, $\mathbf{Pr}[E] \geq 1 - (\epsilon/s)^{2c}$.

Assuming that E holds, then,

2. *There is a variable $x_{i_{j+1}}$ that appears in $1/(2r)$ fraction of the monomials in H_j.*

3. *If $\xi_{j+1} = 0$, then $|H_{j+1}| \leq (1 - 1/(2r))|H_j|$.*

4. *If $\xi_{j+1} = 1$, then $|H_{j+1}| \leq |H_j|$.*

Proof. Let $T = F + G + W$ such that $W = T^{(16r',s]}$. We first show that with probability at least $1 - s(\epsilon/s)^{3c}$, we have $(W_{|q_j})^{[r']} = 0$. Consider $N_1, N_2, \ldots, N_s$ in Lemma 17. Every monomial in W is of the form MN_i for some monomial M and $i \in [s]$. We also have

$|N_i| \ge 4r'$ for all $i \in [s]$. The probability that for some $i \in [s]$ we have that $(N_i)_{|q_j}$ is not zero and of size at most r' is at most $s2^{-3r'} \le s(\epsilon/s)^{3c}$. Therefore, with probability at least $1 - s(\epsilon/s)^{3c}$, we have $(W_{|q_j})^{[r']} = 0$.

By induction, we prove that:

1. $H_j \cap L_j = \varnothing$.
2. H_j contains monomials of size at most r' and
3.

$$\Sigma H_j + \Sigma L_j = F_{|q_j} + G_{|q_j}. \tag{5}$$

For $j = 0$ the result follows from the fact that $H_0 = \mathcal{M}(T^{[r']}) = \mathcal{M}(F)$, $L_0 = \mathcal{M}(T^{(r',16r']}) = \mathcal{M}(G)$ and $q_0 = ()$ is the empty sequence. Assume the above hold for $j-1$. We now prove it for j. 1 and 2 follow immediately from steps (13) and (14) in the algorithm. For 3, we have[4]

$$\begin{aligned} H_j + L_j = H_j \cup L_j &= \mathcal{M}((\Sigma H_{j-1})_{|x_{i_j} \leftarrow \xi_j}) + \mathcal{M}((\Sigma L_{j-1})_{|x_{i_j} \leftarrow \xi_j}) \\ &= \mathcal{M}((\Sigma H_{j-1})_{|x_{i_j} \leftarrow \xi_j} + (\Sigma L_{j-1})_{|x_{i_j} \leftarrow \xi_j}) = \mathcal{M}((\Sigma H_{j-1} + \Sigma L_{j-1})_{|x_{i_j} \leftarrow \xi_j}) \\ &= \mathcal{M}((F_{|q_{j-1}} + G_{|q_{j-1}})_{|x_{i_j} \leftarrow \xi_j}) = \mathcal{M}(F_{|q_j} + G_{|q_j}), \end{aligned}$$

which implies the result. Since H_j contains the monomials of size at most r', $(F_{|q_j})^{[r']} = F_{|q_j}$ and $H_j \cap L_j = \varnothing$, we get $H_j \subseteq (H_j + L_j)^{[r']} = S\left(F_{|q_j} + (G_{|q_j})^{[r']}\right)$. Then, with probability at least $1 - s(\epsilon/s)^{3c}$, we have

$$\begin{aligned} (T_{|q_j})^{[r']} &= (F_{|q_j})^{[r']} + (G_{|q_j})^{[r']} + (W_{|q_j})^{[r']} = F_{|q_j} + (G_{|q_j})^{[r']} + (W_{|q_j})^{[r']} \\ &= F_{|q_j} + (G_{|q_j})^{[r']}, \end{aligned}$$

and then, $H_j \subseteq S\left(F_{|q_j} + (G_{|q_j})^{[r']}\right) = S\left((T_{|q_j})^{[r']}\right)$. This completes the proof of 1.

By Lemma 11, Equation (2) and case 1 of this Lemma, we have: $cd(\Sigma H_j) \le cd(T_{|q_j}^{[r']}) \le cd(T_{|q_j}) \le 2r$. Therefore, by Lemma 12 the result 2 follows.

We now prove (3-4). Since $H_{j+1} = \mathcal{M}((\Sigma H_j)_{|x_{i_{j+1}} \leftarrow \xi_{j+1}}) \backslash \mathcal{M}((\Sigma L_j)_{|x_{i_{j+1}} \leftarrow \xi_{j+1}}) \subseteq \mathcal{M}((\Sigma H_j)_{|x_{i_{j+1}} \leftarrow \xi_{j+1}})$, we get 4. Since $x_{i_{j+1}}$ appears in more than $1/(2r)$ fraction of the monomials in H_j, we get 3. ◀

Before we prove the next result, we give some more notations. For a set of monomials A and $q = (x_{i_1} \leftarrow \xi_1, \ldots, x_{i_j} \leftarrow \xi_j)$, we denote $A_{|q} = \{M_q | M \in A\}$. Recall that $\vee A = \vee_{M \in A} M$. The following properties are easy to prove: Let g be a Boolean function, A, B sets of monomials, $q = (x_{i_1} \leftarrow \xi_1, \ldots, x_{i_j} \leftarrow \xi_j)$ and $q' = (x_{i'_1} \leftarrow \xi'_1, \ldots, x_{i'_j} \leftarrow \xi'_j)$. Then:(I) $(A_{|q})_{|q'} = A_{|q,q'}$, (II) $\mathcal{M}(g_{|q}) \subseteq \mathcal{M}(g)_{|q}$, (III) $(\vee A)_{|q} = \vee A_{|q}$, (IV) if $A \subseteq B$ then[5] $\vee A \Rightarrow \vee B$ and (V) $\Sigma A \Rightarrow \vee A$ and $g \Rightarrow \vee \mathcal{M}(g)$.

Using the above notations and results we prove:

▶ **Lemma 20.** *Suppose events $E_2(c)$ and $E_2(16c)$ in (3) hold. If $\Sigma H_j = \eta$ is a constant function, $\eta \in \{0,1\}$, then with probability at least $1 - (\epsilon/s)^{2c-1}$, $\boldsymbol{Pr}[T_{|q_j} \ne \eta] \le s\left(\frac{\epsilon}{s}\right)^{2c-1}$.*

[4] The operation + for sets is the symmetric difference of sets.

[5] $f \Rightarrow g$ means if $f(x) = 1$ then $g(x) = 1$. In particular, $\mathbf{Pr}[f = 1] \le \mathbf{Pr}[g = 1]$.

Proof. Let $T = F + G + W$, where $F = T^{[r']}$, $G = T^{(r',16r']}$ and $W = T^{(16r',s]}$. We have,

$$\begin{aligned}\mathbf{Pr}[T_{|q_j} \neq \eta] = {} & \mathbf{Pr}[F_{|q_j} + G_{|q_j} + W_{|q_j} \neq \eta] = \mathbf{Pr}[\Sigma H_j + \Sigma L_j + W_{|q_j} \neq \eta] \qquad \text{By (5)}\\ = {} & \mathbf{Pr}[\Sigma L_j + W_{|q_j} \neq 0] \leq \mathbf{Pr}[\Sigma L_j = 1] + \mathbf{Pr}[W_{|q_j} = 1].\end{aligned}$$

Since $W = T^{(16r',s]} \Rightarrow \vee\mathcal{M}(T^{(16r',s]})$, by Lemma 18, we have, with probability at least $1 - s(\epsilon/s)^{32c}$, $\mathbf{Pr}[W_{|q_j} = 1] \leq s\left(\frac{\epsilon}{s}\right)^{32c}$. Since

$$\begin{aligned}L_j &= \mathcal{M}((\Sigma L_{j-1})_{|x_{i_j}\leftarrow\xi_j}) \backslash \mathcal{M}((\Sigma H_{j-1})_{|x_{i_j}\leftarrow\xi_j}) \subseteq \mathcal{M}((\Sigma L_{j-1})_{|x_{i_j}\leftarrow\xi_j})\\ &\subseteq \mathcal{M}(\Sigma L_{j-1})_{|x_{i_j}\leftarrow\xi_j} = (L_{j-1})_{|x_{i_j}\leftarrow\xi_j},\end{aligned}$$

we can conclude that, $L_j \subseteq (L_0)_{|q} = \mathcal{M}(G)_{|q} \subseteq \mathcal{M}(T^{(r',s]})_{|q} = \mathcal{M}(T^{(16cr,s]})_{|q}$, which implies that $\Sigma L_j \Rightarrow \vee L_j \Rightarrow \vee\mathcal{M}(T^{(16cr,s]})_{|q} = (\vee\mathcal{M}(T^{(16cr,s]}))_{|q}$. Therefore, by Lemma 18, with probability at least $1 - s(\epsilon/s)^{2c}$, $\mathbf{Pr}[\Sigma L_j = 1] \leq \mathbf{Pr}[(\vee\mathcal{M}(T^{(16cr,s]}))_{|q}] \leq s\left(\frac{\epsilon}{s}\right)^{2c}$. ◀

▶ **Lemma 21.** *Suppose the events $E_2(4c)$, $E_2(16c)$ and E hold. If f is a size-s decision tree then, with probability at least $1 - (\epsilon/s)^{O(s/\epsilon)}$, ΣH_k is constant for $k \leq 2^{10}\log^2(s/\epsilon)$.*

Proof. By Lemma 16, we have $|H_0| = |\mathcal{M}(F)| = |\mathcal{M}(T^{[r']})| \leq s(s/\epsilon)^{64c}$. By Lemma 19, we have that with probability 1/2, $\xi_j = 1$ and then $|H_{j+1}| \leq |H_j|$. And, with probability 1/2, $\xi_j = 0$ and then $|H_{j+1}| \leq (1 - 1/(2r))|H_j|$. Therefore, when $\xi_1, \ldots, \xi_t$ contains $2r\ln(s(s/\epsilon)^{64c}) \leq 2^8 c\log^2(s/\epsilon)$ zeros, then ΣH_k will be constant for $k \leq t$. The probability of $\xi_i = 0$ is 1/2, and thus, by Chernoff bound the result follows. ◀

▶ **Lemma 22.** *If f is a size-s decision tree, then with probability at least $1 - poly(\epsilon/s)$, the tester accepts.*

Proof. By Lemma 21, with probability at least $1 - poly(\epsilon/s)$, the tester does not reject inside the Repeat loop. By Lemma 20, with probability at least $1 - poly(\epsilon/s)$, $\mathbf{Pr}[T_{|q_j} \neq \eta] \leq poly(\epsilon/s)$, hence, with probability at least $1 - poly(\epsilon/s)$, the tester will not reject in line 20. ◀

▶ **Lemma 23.** *Let $R = 2^{10}c\log^2(s/\epsilon)$. If f is ϵ-far from every size-2^R decision tree, then with probability at least 2/3, the tester rejects.*

Proof. If f is ϵ-far from every size-2^R decision tree, then $T = f(x + a)$ is ϵ-far from every size-2^R decision tree.

Consider the tree T^* that is generated in the tester for all possible random walks, where each node in the tree is labeled with the variable x_{i_j} that appears in at least $1/(2r)$ of the monomials in H_{j-1} if such variable exists, and is labeled with Reject when the algorithm reaches Reject. In the tree, we will have three types of nodes that are labeled with Reject. Type I are nodes where there is no variable that appears in at least $1/(2r)$ fractions of the monomials of H_{j-1}. Type II are the nodes that are of depth $R + 1$, and Type III are the nodes of depth less than R where ΣH_j is constant η and $\mathbf{Pr}[T_{|q_j} = \eta] \geq \epsilon/4$.

If, with probability at least $\epsilon/4$, a random walk in the tree T^* reaches a Reject node, then the probability that the tester rejects is $1 - \left(1 - \frac{\epsilon}{4}\right)^{40/\epsilon} \geq \frac{2}{3}$, and we are done.

Suppose, for the contrary, this is not true. Then, define a decision tree T' that is equal to T^*, where each Reject node is replaced with a leaf labelled with 0, and each other other leaf is labeled with 0 if $\mathbf{Pr}[T_{|q_j}] \leq \epsilon/4$ and 1 if $\mathbf{Pr}[T_{|q_j}] \geq 1 - \epsilon/4$. Then, T' is a depth-R tree (and therefore size-2^R tree). The probability that $T'(x)$ is not equal to $T(x)$ is less than the probability that a random walk arrives to a Reject leaf, or if it arrives to a non-Reject leaf, then $T_{|q_j}(x)$ is not equal to the label in the leaf. Therefore, $\mathbf{Pr}[T'(x) \neq T(x)] \leq \frac{\epsilon}{4} + \frac{\epsilon}{4} < \epsilon$, a contradiction. ◀

As we said earlier, the above tester runs in $poly(n, 1/\epsilon)$ time and queries. We now use the reduction of Bshouty in [6] and get a tester that runs in time $poly(s, 1/\epsilon)n$ and makes $poly(s, 1/\epsilon)$ queries. See Lemma 26 in Appendix A.

References

1 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 210–220, 2011. doi:10.1109/CCC.2011.31.

2 Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. *CoRR*, abs/2109.00637, 2021. arXiv:2109.00637.

3 Guy Blanc, Jane Lange, and Li-Yang Tan. Testing and reconstruction via decision trees. *CoRR*, abs/2012.08735, 2020. arXiv:2012.08735.

4 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.

5 Nader H. Bshouty. Almost optimal testers for concise representations. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:156, 2019. URL: https://eccc.weizmann.ac.il/report/2019/156.

6 Nader H. Bshouty. Almost optimal testers for concise representations. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 5:1–5:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.5.

7 Nader H. Bshouty and Catherine A. Haddad-Zaknoon. Adaptive exact learning of decision trees from membership queries. In Aurélien Garivier and Satyen Kale, editors, *Algorithmic Learning Theory, ALT 2019, 22-24 March 2019, Chicago, Illinois, USA*, volume 98 of *Proceedings of Machine Learning Research*, pages 207–234. PMLR, 2019. URL: http://proceedings.mlr.press/v98/bshouty19a.html.

8 Nader H. Bshouty and Yishay Mansour. Simple learning algorithms for decision trees and multivariate polynomials. *SIAM J. Comput.*, 31(6):1909–1925, 2002. doi:10.1137/S009753979732058X.

9 Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 545–556, 2011. doi:10.1007/978-3-642-22006-7_46.

10 Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 549–558, 2007. doi:10.1109/FOCS.2007.32.

11 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

12 Michael J. Kearns and Dana Ron. Testing problems with sublearning sample complexity. *J. Comput. Syst. Sci.*, 61(3):428–456, 2000. doi:10.1006/jcss.1999.1656.

13 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

A Reductions in Testing

The following reductions are deduced immediately from [5, 6].

We say that a class C is *closed under zero-one projection* if for every $f \in C$, every $i \in [n]$ and every $\xi \in \{0,1\}$ we have $f_{|x_i \leftarrow \xi} \in C$. We say that C is *symmetric* if for every permutation $\pi : [n] \to [n]$ and every $f \in C$ we have $f_\pi \in C$ where $f_\pi(x) := f(x_{\pi(1)}, \cdots, x_{\pi(n)})$. All the classes in this paper are closed under zero-one projection and symmetric.

The following results are proved in [6] (Theorem 2) when $H = C$. The same proof works for any $C \subseteq H$.

▶ **Lemma 24** ([6]). *Let $C \subseteq H$ be classes of Boolean functions (over n variables) that are symmetric sub-classes of k-$\mathbb{JUNTA}$ and are closed under zero-one projection. Suppose C is distribution-free learnable from H in time $T(n,\epsilon,\delta)$ using $Q(n,\epsilon,\delta)$ random example queries and $M(n,\epsilon,\delta)$ black-box queries. Then, there is a distribution-free two-sided adaptive algorithm for ϵ-testing C by H that runs in time $T(k,\epsilon/12,1/24) + O(mn)$ and makes $m = \tilde{O}\left(M(k,\epsilon/12,1/24) + k \cdot Q(k,\epsilon/12,1/24) + \frac{k}{\epsilon}\right)$ queries.*

▶ **Lemma 25** ([6]). *Let $C \subseteq H$ be classes of Boolean functions (over n variables) that are symmetric sub-classes of k-$\mathbb{JUNTA}$ and are closed under zero-one projection. Suppose C is learnable from H under the uniform distribution in time $T(n,\epsilon,\delta)$ using $Q(n,\epsilon,\delta)$ random example queries and $M(n,\epsilon,\delta)$ black-box queries. Then, there is a (uniform distribution) two-sided adaptive algorithm for ϵ-testing C by H runs in time $T(k,\epsilon/12,1/24) + O(mn)$ and that makes $m = \tilde{O}\left(M(k,\epsilon/12,1/24) + Q(k,\epsilon/12,1/24) + \frac{k}{\epsilon}\right)$ queries.*

The following is proved in [6] when $H = C$. The same proof gives the following result:

▶ **Lemma 26** ([6]). *Let C and $C \subseteq H$ be classes of Boolean functions that are symmetric sub-classes of k-$\mathbb{JUNTA}$ and are closed under zero-one projection. Suppose there is a tester $\mathcal{T}$ for $C_k = \{f \in C | f$ is independent on $x_{k+1}, \ldots, x_n\}$ such that*

1. *$\mathcal{T}$ is a two-sided adaptive ϵ-tester (resp. distribution-free ϵ-tester) that runs in time $T(k,\epsilon,\delta)$.*
2. *If $f \in C_k$ then, with probability at least $1-\delta$, $\mathcal{T}$ accepts.*
3. *If f is ϵ-far from every function in H_k (resp., with respect to $\mathcal{D}$) then, with probability at least $1-\delta$, $\mathcal{T}$ rejects.*
4. *$\mathcal{T}$ makes $Q(k,\epsilon,\delta)$ random example queries and $M(k,\epsilon,\delta)$ black-box queries.*

Then, there is a two-sided adaptive algorithm for ϵ-testing (resp., distribution-free algorithm for ϵ-testing) C by H that makes $m = \tilde{O}\left(M(k,\epsilon/12,1/24) + Q(k,\epsilon/12,1/24) + \frac{k}{\epsilon}\right)$ (resp., makes $m = \tilde{O}\left(M(k,\epsilon/12,1/24) + k \cdot Q(k,\epsilon/12,1/24) + \frac{k}{\epsilon}\right)$) queries and runs in time $T(k,\epsilon/12,1/24) + O(mn)$.

The Ideal Membership Problem and Abelian Groups

Andrei A. Bulatov
School of Computing Science, Simon Fraser University, Burnaby, Canada

Akbar Rafiey
School of Computing Science, Simon Fraser University, Burnaby, Canada

Abstract

Given polynomials $f_0, f_1, \dots, f_k$ the Ideal Membership Problem, IMP for short, asks if f_0 belongs to the ideal generated by $f_1, \dots, f_k$. In the search version of this problem the task is to find a proof of this fact. The IMP is a well-known fundamental problem with numerous applications, for instance, it underlies many proof systems based on polynomials such as Nullstellensatz, Polynomial Calculus, and Sum-of-Squares. Although the IMP is in general intractable, in many important cases it can be efficiently solved.

Mastrolilli [SODA'19] initiated a systematic study of IMPs for ideals arising from Constraint Satisfaction Problems (CSPs), parameterized by constraint languages, denoted IMP(Γ). The ultimate goal of this line of research is to classify all such IMPs accordingly to their complexity. Mastrolilli achieved this goal for IMPs arising from CSP(Γ) where Γ is a Boolean constraint language, while Bulatov and Rafiey [arXiv'21] advanced these results to several cases of CSPs over finite domains. In this paper we consider IMPs arising from CSPs over "affine" constraint languages, in which constraints are subgroups (or their cosets) of direct products of Abelian groups. This kind of CSPs include systems of linear equations and are considered one of the most important types of tractable CSPs. Some special cases of the problem have been considered before by Bharathi and Mastrolilli [MFCS'21] for linear equation modulo 2, and by Bulatov and Rafiey [arXiv'21] to systems of linear equations over $\mathsf{GF}(p)$, p prime. Here we prove that if Γ is an affine constraint language then IMP(Γ) is solvable in polynomial time assuming the input polynomial has bounded degree.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems; Mathematics of computing → Gröbner bases and other special bases

Keywords and phrases Polynomial Ideal Membership, Constraint Satisfaction Problems, Polymorphisms, Gröbner Bases, Abelian Groups

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.18

Related Version *Full Version*: https://arxiv.org/pdf/2201.05218.pdf [16]

Funding *Andrei A. Bulatov*: Research supported by an NSERC Discovery Grant.
Akbar Rafiey: Research supported by NSERC.

1 Introduction

The Ideal Membership Problem. Representing combinatorial problems by polynomials and then using algebraic techniques to approach them is one of the standard methods in algorithms and complexity. The Ideal Membership Problem (IMP for short) is an important algebraic framework that has been instrumental in such an approach. The IMP underlies many proof systems based on polynomials such as Nullstellensatz, Polynomial Calculus, and Sum-of-Squares, and therefore plays an important role in such areas as proof complexity and approximation.

Let $\mathbb{F}$ be a field and $\mathbb{F}[x_1, \dots, x_n]$ the ring of polynomials over $\mathbb{F}$. Given polynomials f_0, $f_1, \dots, f_k \in \mathbb{F}[x_1, \dots, x_n]$ the IMP asks if f_0 belongs to the ideal $\langle f_1, \dots, f_k\rangle$ generated by $f_1, \dots, f_k$. This fact is usually demonstrated by presenting a *proof*, that is, a collection of

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 18; pp. 18:1–18:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

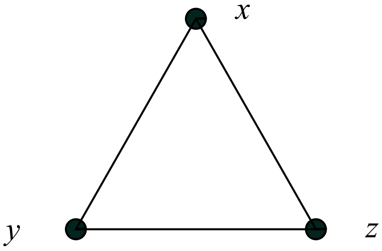


Figure 1 Graph 2-colorability.

polynomials $h_1, \ldots, h_k$ such that the following polynomial identity holds $f_0 = h_1 f_1 + \cdots + h_k f_k$. Many applications require the ability to produce such a proof. We refer to the problem of finding a proof of membership as the *search IMP*. Note that by the Hilbert Basis Theorem any ideal of $\mathbb{F}[x_1, \ldots, x_n]$ can be represented by a finite set of generators meaning that the above formulation of the problem covers all possible ideals of $\mathbb{F}[x_1, \ldots, x_n]$.

The general IMP is a difficult problem and it is not even obvious whether or not it is decidable. The decidability was established in [23, 32, 33]. Then Mayr and Meyer [29] were the first to study the complexity of the IMP. They proved an exponential space lower bound for the membership problem for ideals generated by polynomials with integer and rational coefficients. Mayer [28] went on establishing an exponential space upper bound for the IMP for ideals over $\mathbb{Q}$, thus proving that such IMPs are **EXPSPACE**-complete. The source of hardness here is that a proof that $f_0 \in \langle f_1, \ldots, f_k \rangle$ may require polynomials of exponential degree. In the cases when the degree of a proof has a linear bound in the degree of f_0, the IMP can be solved more efficiently. (There is also the issue of exponentially long coefficients that we will mention later.)

Combinatorial Ideals. To illustrate the connection of the IMP to combinatorial problems we consider the following simple example. We claim that the graph in Fig. 1 is 2-colorable if and only if polynomials $x(1-x), y(1-y), z(1-z), x+y-1, x+z-1, y+z-1$ have a common zero. Indeed, denoting the two possible colors 0 and 1, the first three polynomials guarantee that the only zeroes this collection of polynomials can have are such that $x, y, z \in \{0, 1\}$. Then the last three polynomials make sure that in every common zero the values of x, y, z are pairwise different, and so correspond to a proper coloring of the graph. Of course, the graph in the picture is not 2-colorable, and by the Weak Nullstellensatz this is so if and only if the constant polynomial 1 belongs to the ideal generated by the polynomials above. A proof of that can be easily found

$$1 = (-4)\,[x(x-1)] + (2x-1)\,([x+y-1] - [y+z-1] + [x+z-1])\,.$$

The example above exploits the connection between polynomial ideals and sets of zeroes of polynomials, also known as *affine varieties*. While this connection does not necessarily holds in the general case, as Hilbert's Nullstellensatz requires certain additional properties of ideals, it works for so called *combinatorial ideals* that arise from the majority of combinatorial problems similar to the example above. The varieties corresponding to combinatorial ideals are finite, and the ideals themselves are zero-dimensional and radical. These properties make the IMP significantly easier, in particular, it can be solved in single exponential time [20]. Also, Hilbert's Strong Nullstellensatz holds in this case, which means that if the IMP is restricted to radical ideals, it is equivalent to (negation of) the question: given $f_0, f_1, \ldots, f_k$ does there exist a zero of $f_1, \ldots, f_k$ that is not a zero of f_0.

The special case of the IMP with $f_0 = 1$ has been studied for combinatorial problems in the context of lower bounds on Polynomial Calculus and Nullstellensatz proofs, see e.g. [4, 17, 22]. A broader approach of using polynomials to represent finite-domain constraints has been explored in [18, 26]. Clegg et al., [18], discuss a propositional proof system based on a bounded degree version of Buchberger's algorithm [9] for finding proofs of unsatisfiability. Jefferson et al., [26] use a modified form of Buchberger's algorithm that can be used to achieve the same benefits as the local-consistency algorithms which are widely used in constraint processing.

Applications in other proof systems. The bit complexity of various (semi)algebraic proof systems is another link that connects approximation algorithms and the IMP. As is easily seen, if the degree of polynomials $h_1, \dots, h_k$ in a proof $f_0 = h_1 f_1 + \dots + h_k f_k$ is bounded, their coefficients can be found by representing this identity through a system of linear equations. A similar approach is used in other (semi)algebraic proof systems such as Sum-of-Squares (SOS), in which bounded degree proofs can be expressed through an SDP program. Thus, if in addition to low degree the system of linear equations or the SDP program has a solution that can be represented with a polynomial number of bits (thus having low *bit complexity*), a proof can be efficiently found.

However, O'Donnell [30] proved that low degree of proofs does not necessarily imply its low bit complexity. He presented a collection of polynomials that admit bounded degree SOS proofs of nonnegativity, all such proofs involve polynomials with coefficients of exponential length. This means that the standard methods of solving SDPs such as the Ellipsoid Method would take exponential time to complete. Raghavendra and Weitz [31] suggested some sufficient conditions on combinatorial ideals that guarantee a low bit complexity SOS proof exists whenever a low degree one does. Two of these conditions hold for the majority of combinatorial problems, and the third one is so called k-effectiveness of the IMP part of the proof. In [15], we showed that for problems where the IMP part is of the form IMP(Γ) (to be introduced shortly) only one of the first two conditions remains somewhat nontrivial and k-effectiveness can be replaced with the requirement that a variation of IMP(Γ) is solvable in polynomial time.

The IMP and the CSP. In this paper we consider IMPs that arise from a specific class of combinatorial problems, the Constraint Satisfaction Problem or the CSP for short. In a CSP we are given a set of variables, and a collection of constraints on the values that variables are allowed to be assigned simultaneously. The question in a CSP is whether there is an assignment to variables that satisfies all the constraints. The CSP provides a general framework for a wide variety of combinatorial problems, and it is therefore very natural to study the IMPs that arise from constraint satisfaction problems.

One of the major directions in the CSP research is the study of CSPs in which the allowed types of constraints are restricted. Such restrictions are usually represented by a *constraint language* that is a set of relations or predicates on a fixed set. The CSP parametrized by a constraint language Γ is denoted CSP(Γ).

Mastrolilli in [27] initiated a systematic study of IMPs that arise from problems of the form CSP(Γ), denoted IMP(Γ). More precisely, for a constraint language Γ over domain $D = \{0, \dots, d-1\} \subseteq \mathbb{F}$, in an instance of IMP($\Gamma$) we are given an instance $\mathcal{P}$ of CSP(Γ) with set of variables $X = \{x_1, \dots, x_n\}$, and a polynomial $f_0 \in \mathbb{F}[x_1, \dots, x_n]$. The question is whether or not f_0 belongs to the ideal $\mathcal{I}(\mathcal{P})$ of $\mathbb{F}[x_1, \dots, x_n]$, where the corresponding variety of $\mathcal{I}(\mathcal{P})$ equals the set of solutions of $\mathcal{P}$. Observe, that using Hilbert's Strong Nullstellensatz the problem can also be reformulated as, whether there exists a solution to $\mathcal{P}$ that is not a zero of f_0. Sometimes we need to restrict the degree of the input polynomial, the IMP in which the degree of f_0 is bounded by d is denoted by $\mathrm{IMP}_d(\Gamma)$.

The complexity of the IMP. The main research question considered in [27] is to classify the problems $\mathrm{IMP}(\Gamma)$ according to their complexity. We [15] showed that in all known cases $\mathrm{IMP}_d(\Gamma)$ can be solved in polynomial time (for any fixed d) if and only if a Gröbner Basis can be efficiently constructed.

Mastrolilli [27] along with Mastrolilli and Bharathi [6] succeeded in characterizing the complexity of $\mathrm{IMP}_d(\Gamma)$ for constraint languages Γ over a 2-element domain. Their results are best presented using the language of polymorphisms. Recall that a *polymorphism* of a constraint language Γ over a set D is a multi-ary operation on D that can be viewed as a multi-dimensional symmetry of relations from Γ. By $\mathsf{Pol}(\Gamma)$ we denote the set of all polymorphisms of Γ. As for the CSP, polymorphisms of Γ is what determines the complexity of $\mathrm{IMP}(\Gamma)$, see [15].

According to [27, 6], let Γ be a constraint language over $D = \{0, 1\}$ such that the *constant relations* $R_0, R_1 \in \Gamma$, where $R_i = \{(i)\}$. Then $\mathrm{IMP}_d(\Gamma)$ is polynomial time solvable if Γ is invariant under a semilattice or affine operation (of $\mathbb{Z}_2$), the problem $\mathrm{IMP}(\Gamma)$ is polynomial time solvable if Γ is invariant under a majority polymorphism. Otherwise $\mathrm{IMP}_0(\Gamma)$ is **coNP**-complete. This result has been improved in [15] (see also [5, 7]) by showing that $\mathrm{IMP}_d(\Gamma)$ remains polynomial time when Γ has an arbitrary semilattice polymorphism, not only on a 2-element set, an arbitrary dual-discriminator polymorphism, or an affine polymorphism of $\mathbb{Z}_p$, p prime.

Solving the IMP. The IMP is mostly solved using one of the two methods. The first one is the method of finding an IMP or SOS proofs of bounded degree using systems of linear equations or SDP programs. The other approach uses *Gröbner bases* and the standard polynomial division to verify whether a given polynomial has zero remainder when divided by generators of an ideal: if this is the case, the polynomial belongs to the ideal. However, constructing a Gröbner basis is not always feasible, as even though the original generating set is small, the corresponding Gröbner basis may be huge. Note however that to solve the IMP_d it suffices to construct a degree d Gröbner Basis, a.k.a d-truncated Gröbner Basis.

A more sophisticated approach was suggested in [15]. It involves reductions between problems of the form $\mathrm{IMP}(\Gamma)$ before arriving to one for which a Gröbner basis can be constructed in a relatively simple way. Moreover, [15] also introduces a slightly different form of the IMP, called the χIMP, in which the input polynomial has indeterminates as some of its coefficients, and the problem is to find values for those indeterminates (if they exist) such that the resulting polynomial belongs to the given ideal. We showed that χIMP is solvable in polynomial time for every known case of polynomial time solvable IMP, and that χIMP helps to solve the search version of the IMP.

▶ **Theorem 1** ([15]).

(1) *If Γ has a semilattice, dual-discriminator, or the affine polymorphism of $\mathbb{Z}_p$, p prime, then $\chi IMP_d(\Gamma)$ is solvable in polynomial time for every d.*

(2) *If $\chi IMP_d(\Gamma)$ is polynomial time solvable then for every instance $\mathcal{P}$ of CSP(Γ) a d-truncated Gröbner basis of $\mathtt{I}(\mathcal{P})$ can be found in polynomial time.*

Our contribution

Affine operations. In this paper we consider IMPs over languages invariant under affine operations of arbitrary finite Abelian groups. This type of constraint languages played an important role in the study of the CSP for three reasons. First, it captures a very natural class of problems. Problems CSP(Γ) where Γ is invariant under an affine operation of a finite field $\mathbb{F}$ can be expressed by systems of linear equations over $\mathbb{F}$ and therefore admit a classic solution algorithm such as Gaussian elimination or coset generation. In the case of a general Abelian group $\mathbb{A}$ the connection with systems of linear equations is more complicated, although it is still true that every instance of CSP(Γ) in this case can be thought of as a system of linear equations with coefficients from some ring – the ring of endomorphisms of $\mathbb{A}$.

Second, it has been observed that there are two main algorithmic approaches to solving the CSP. The first one is based on the local consistency of the problem. CSPs that can be solved solely by establishing some kind of local consistency are said to have *bounded width* [14, 2]. The property to have bounded width is related to a rather surprising number of other seemingly unrelated properties, see e.g. [1, 34]. CSP algorithms of the second type are based on the *few subalgebras* property and achieve results similar to those of Gaussian elimination: they construct a concise representation of the set of all solutions of a CSP [11, 24]. Problems CSP(Γ) where Γ has an affine polymorphism were pivotal in the development of few subpowers algorithms, and, in a sense, constitute the main nontrivial case of them. Among the existing results on the IMP, IMP(Γ) for Γ invariant under a semilattice or majority polymorphism belong to the local consistency part of the algorithmic spectrum, while those for Γ invariant with respect to an affine operation are on the “few subalgebras” part of it. It is therefore important to observe differences in approaches to the IMP in these two cases.

Third, the few subalgebras algorithms [11, 24] when applied to systems of linear equations serve as an alternative to Gaussian elimination that also work in a more general situation and are less sensitive to the algebraic structure behind the problem. There is, therefore, a hope that studying IMPs with an affine polymorphism may teach us about proof systems that use the IMP and do not quite work in the affine case.

The main result of this paper is

▶ **Theorem 2.** *Let $\mathbb{A}$ be an Abelian group and Γ a constraint language such that the affine operation $x - y + z$ of $\mathbb{A}$ is a polymorphism of Γ. Then IMP$_d$(Γ) can be solved in polynomial time for any d. Moreover, given an instance $(f_0, \mathcal{P})$ of IMP$_d$(Γ) a (d-truncated) Gröbner basis of $\mathtt{I}(\mathcal{P})$ can be constructed in polynomial time.*

The tractability of affine IMPs. In [6, 7, 15, 27] IMPs invariant under an affine polymorphism are represented as systems of linear equations that are first transformed to a reduced row-echelon form using Gaussian elimination, and then further converted into a Gröbner basis of the corresponding ideal. If Γ is a constraint language invariant under the affine operation of a general Abelian group $\mathbb{A}$, none of these three steps work: an instance generally cannot be represented as a system of linear equations, Gaussian elimination does not work on systems of linear equations over an arbitrary Abelian group, and a reduced row-echelon form cannot be converted into a Gröbner basis. We therefore need to use a completely different approach, see Section 4. Given an instance $(f_0, \mathcal{P})$ of IMP(Γ) we use the Fundamental Theorem of Abelian groups and a generalized version of pp-interpretations for the IMP [15] to reduce $(f_0, \mathcal{P})$ to an instance $(f'_0, \mathcal{P}')$ of *multi-sorted* IMP(Δ) (see below), in which every variable takes values from a set of the form $\mathbb{Z}_{p^\ell}$, p prime. Then we replace the domains $\mathbb{Z}_{p^\ell}$ of $(f_0', \mathcal{P}')$ by sets of roots of unity that allows for a more concise representation of polynomials. Finally, we show that a (truncated) Gröbner Basis for the resulting problem can be efficiently constructed.

Multi-sorted IMPs. In order to prove Theorem 2 we introduce two techniques new to the IMP research, although the first one has been extensively used for the CSP. The first technique is multi-sorted problems mentioned above, where every variable has its own domain of values. This framework is standard for the CSP, and also works very well for the IMP, as long as the domain of each variable can be embedded into the field of real or complex numbers. However, many concepts used in proofs and solution algorithms such as pp-definitions, pp-interpretations, polymorphisms have to be significantly adjusted, and several existing results have to be reproved in this more general setting. However, in spite of this extra work, the multi-sorted IMP may become the standard framework in this line of research.

A general approach to χIMP. In [15] we introduced χIMP, a variation of the IMP, in which given a CSP instance $\mathcal{P}$ and a polynomial f_0 some of whose coefficients are unknown, the goal is to find values of the unknown coefficients such that the resulting polynomial f_0' belongs to $\mathtt{I}(\mathcal{P})$; or report such values do not exist. This framework has been instrumental in finding a Gröbner basis and therefore solving the search version of the IMPs mentioned earlier, as well as in establishing connections between the IMP and other proof systems such as SOS. We again use χIMP to prove the second part of Theorem 2. In order to do that we improve the approach in two ways. First, we adapt it for multi-sorted problems. Second, while in [15] reductions for χIMP are proved in an ad hoc manner, here we develop a unifying construction based on substitution reductions that covers all the useful cases so far.

2 Preliminaries

Ideals and varieties. We follow the same notation and terminology as [15, 19, 27]. Let $\mathbb{F}$ denote an arbitrary field and $\mathbb{F}[x_1, \dots, x_n]$ be the ring of polynomials over the field $\mathbb{F}$ and indeterminates $x_1, \dots, x_n$. Sometimes it will be convenient not to assume any specific ordering or names of the indeterminates. In such cases we use $\mathbb{F}[X]$, where X is a set of indeterminates, and treat points in $\mathbb{F}^X$ as mappings $\varphi : X \to \mathbb{F}$. The value of a polynomial $f \in \mathbb{F}[X]$ is then written as $f(\varphi)$. The *ideal* of $\mathbb{F}[x_1, \dots, x_n]$ generated by a finite set of polynomials $\{f_1, \dots, f_m\}$ in $\mathbb{F}[x_1, \dots, x_n]$ is defined as $\langle f_1, \dots, f_m \rangle \overset{\text{def}}{=} \left\{ \sum_{i=1}^{m} t_i f_i \mid t_i \in \mathbb{F}[x_1, \dots, x_n] \right\}$. For a set of points $S \subseteq \mathbb{F}^n$ its *vanishing ideal* is the set of polynomials defined as

$$\mathbf{I}(S) \overset{\text{def}}{=} \{f \in \mathbb{F}[x_1, \dots, x_n] \mid f(a_1, \dots, a_n) = 0 \;\; \forall (a_1, \dots, a_n) \in S\}.$$

For an ideal $\mathtt{I} \subseteq \mathbb{F}[x_1, \dots, x_n]$ its *affine variety* is the set of common zeros of all the polynomials in $\mathtt{I}$. This is denoted by $\mathbf{V}(\mathtt{I})$ and is formally defined as

$$\mathbf{V}(\mathtt{I}) = \{(a_1, \dots, a_n) \in \mathbb{F}^n \mid f(a_1, \dots, a_n) = 0 \quad \forall f \in \mathtt{I}\}.$$

The (multi-sorted) CSP. In the majority of theoretical studies of the CSP all variables are assumed to have the same domain, this type of CSPs are known as *one-sorted CSPs*. However, for various purposes, mainly for more involved algorithms such as in [10, 35] one might consider CSPs where different variables of a CSP have different domains, this type of CSPs are known as *multi-sorted CSPs* [12]. Definitions below are from [12].

▶ **Definition 3.** *For any finite collection of finite domains $\mathcal{D} = \{D_t \mid t \in T\}$, and any list of indices $(t_1, t_2, \ldots, t_m) \in T^m$, a subset R of $D_{t_1} \times D_{t_2} \times \cdots \times D_{t_m}$, together with the list $(t_1, t_2, \ldots, t_m)$, is called a* multi-sorted *relation over $\mathcal{D}$ with arity m and signature $(t_1, t_2, \ldots, t_m)$. For any such relation R, the signature of R is denoted $\sigma(R)$.*

As an example consider $\mathcal{D} = \{D_1, D_2\}$ with $D_1 = \{0,1\}$, $D_2 = \{0,1,2\}$. Then $\mathbb{Z}_6$, which is the direct sum of $\mathbb{Z}_2$ and $\mathbb{Z}_3$, $\mathbb{Z}_2 \oplus \mathbb{Z}_3$, can be viewed as a multi-sorted relation over $\mathcal{D}$ of arity 2 with signature $(1,2)$.

Given any set of multi-sorted relations, we can define a corresponding class of multi-sorted CSPs. Let Γ be a set of multi-sorted relations over a collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$. The multi-sorted constraint satisfaction problem over Γ, denoted MCSP(Γ), is defined to be the decision problem with instance $\mathcal{P} = (X, \mathcal{D}, \delta, \mathcal{C})$, where X is a finite set of variables, $\delta : X \to T$, and $\mathcal{C}$ is a set of constraints where each constraint $C \in \mathcal{C}$ is a pair $\langle \mathbf{s}, R \rangle$, so that

- $\mathbf{s} = (x_1, \ldots, x_{m_C})$ is a tuple of variables of length m_C, called the constraint scope;
- R is from Γ with arity m_C and signature $(\delta(x_1), \ldots, \delta(x_{m_c}))$, called the constraint relation.

The goal is to decide whether or not there exists a solution, i.e. a mapping $\varphi : X \to \cup_{D \in \mathcal{D}} D$, with $\varphi(x) \in D_{\delta(x)}$, satisfying all of the constraints. We will use $\mathsf{Sol}(\mathcal{P})$ to denote the (possibly empty) set of solutions of the instance $\mathcal{P}$.

The ideal-CSP correspondence. For an instance $\mathcal{P} = (X, \mathcal{D}, \delta, \mathcal{C})$ of MCSP(Γ) we wish to map $\mathsf{Sol}(\mathcal{P})$ to an ideal $\mathcal{I}(\mathcal{P}) \subseteq \mathbb{F}[X]$ ($\mathbb{F}$ is supposed to contain $\cup_{D \in \mathcal{D}} D$, and therefore usually is considered to be a numerical field) such that $\mathsf{Sol}(\mathcal{P}) = \mathbf{V}(\mathcal{I}(\mathcal{P}))$. The (radical) ideal $\mathcal{I}(\mathcal{P})$ of $\mathbb{F}[x_1, \ldots, x_n]$ whose corresponding variety equals the set of solutions of $\mathcal{P}$ is constructed as follows. First, for every x_i the ideal $\mathcal{I}(\mathcal{P})$ contains a *domain* polynomial $f_{\mathcal{D}}(x_i) = \prod_{a \in D_{\delta(x_i)}} (x_i - a)$ whose zeroes are precisely the elements of $D_{\delta(x_i)}$ (this ensures that $\mathcal{I}(\mathcal{P})$ is radical). Then for every constraint $R(x_{i_1}, \ldots, x_{i_k})$, where R is a predicate on $\mathcal{D}$, the ideal $\mathcal{I}(\mathcal{P})$ contains a polynomial $f_R(x_{i_1}, \ldots, x_{i_k})$ that interpolates R, that is, for $(x_{i_1}, \ldots, x_{i_k})$ it holds $f_R(x_{i_1}, \ldots, x_{i_k}) = 0$ if and only if $R(x_{i_1}, \ldots, x_{i_k})$ is true. This model generalizes a number of constructions used in the literature to apply Nullstellensatz or SOS proof systems to combinatorial problems, see, e.g., [4, 17, 22, 31]. If $\mathcal{D} = \{D\}$ in the above definitions then we obtain the definitions for the one-sorted CSP and IMP. Moreover, as observed for the one-sorted case [27, 15], due to the presence of domain polynomials we have $\mathbf{V}(\mathcal{I}(\mathcal{P})) = \emptyset \Leftrightarrow 1 \in \mathcal{I}(\mathcal{P}) \Leftrightarrow \mathcal{I}(\mathcal{P}) = \mathbb{F}[X]$.

In the general Ideal Membership Problem we are given an ideal $\mathcal{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$, usually by some finite generating set, and a polynomial f_0. The question then is to decide whether or not $f_0 \in \mathcal{I}$. If $\mathcal{I}$ is given through a CSP instance, we can be more specific.

▶ **Definition 4.** *The* Ideal Membership Problem *associated with a constraint language Γ over a set $\mathcal{D}$ is the problem IMP(Γ) in which the input is a pair $(f_0, \mathcal{P})$ where $\mathcal{P} = (X, \mathcal{D}, \delta, \mathcal{C})$ is a MCSP(Γ) instance and f_0 is a polynomial from $\mathbb{F}[X]$. The goal is to decide whether f_0 lies in the ideal $\mathtt{I}(\mathcal{P})$. We use IMP$_d(\Gamma)$ to denote IMP(Γ) when the input polynomial f_0 has degree at most d.*

We say that IMP(Γ) is *tractable* if it can be solved in polynomial time, and IMP(Γ) is *d-tractable* if IMP$_d(\Gamma)$ can be solved in polynomial time for every d.

IMP and Gröbner Bases. The Gröbner Basis G of an ideal is a set of generators with some particular properties that allow for efficient solving of the IMP. If we restrict ourselves to the polynomials of degree at most d then we obtain a *d-truncated Gröbner Basis.* The d-truncated Gröbner Basis G_d of G is defined as $G_d = G \cap \mathbb{F}[x_1, \ldots, x_n]_d$ where $\mathbb{F}[x_1, \ldots, x_n]_d$

denotes the subset of polynomials of degree at most d. To solve IMP_d it suffices to compute a d-truncated Gröbner Basis. This is because, for the input polynomial f_0 of degree d, the only polynomials from G that can possibly divide f_0 are those from G_d. Moreover, the remainders of such divisions have degree at most d.

3 Multi-sorted CSPs and IMP

We study multi-sorted CSPs in the context of the IMP and provide a reduction for multi-sorted languages that are pp-interpretable. This in particular is useful in this paper as it provides a reduction between languages that are invariant under an affine polymorphism over an arbitrary Abelian group and languages over several cyclic p-groups.

3.1 Primitive-positive definability and interpretability

Primitive-positive (pp-) definitions have proved to be instrumental in the study of the CSP [25, 13] and of the IMP as well [15]. Here we introduce the definition of pp-definitions and the more powerful construction, pp-interpretations, in the multi-sorted case, and prove that, similar to the one-sorted case [15], they give rise to reductions between IMPs.

▶ **Definition 5** (pp-definability). *Let Γ be a multi-sorted constraint language on a collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$. A* primitive-positive (pp-) formula *in the language Γ is a first order formula L over variables X that uses predicates from Γ, equality relations, existential quantifier, and conjunctions, and satisfies the condition:*

- *if $R_1(x_1, \dots, x_k), R_2(y_1, \dots, y_\ell)$ are atomic formulas in L with signatures σ_1, σ_2 and such that x_i, y_j is the same variable, then $\sigma_1(i) = \sigma_2(j)$.*

The condition above determines the signature $\sigma : X \to T$ of L.

Let Δ be another multi-sorted language over $\mathcal{D}$. We say that Γ pp-defines Δ (or Δ is pp-definable from Γ) if for each (k-ary) relation (predicate) $R \in \Delta$ there exists a pp-formula L over variables $\{x_1, \dots, x_m, x_{m+1}, \dots, x_{m+k}\}$ such that

$$R(x_{m+1}, \dots, x_{m+k}) = \exists x_1 \dots \exists x_m L,$$

and if σ, σ' are the signatures of L and R, respectively, then $\sigma' = \sigma_{|_{\{m+1,\dots,m+k\}}}$.

Multi-sorted primitive-positive (pp-) interpretations are also similar to the one-sorted case [15], but require a bit more care.

▶ **Definition 6** (pp-interpretability). *Let Γ, Δ be multi-sorted constraint languages over finite collections of sets $\mathcal{D} = \{D_t \mid t \in T\}, \mathcal{E} = \{E_s \mid s \in S\}$, respectively, and Δ is finite. We say that Γ pp-interprets Δ if for every $s \in S$ there exist $i_{s,1}, \dots, i_{s,\ell_s} \in T$, a set $F_s \subseteq D_{i_{s,1}} \times \dots \times D_{i_{s,\ell_s}}$, and an onto mapping $\pi_s : F_s \to E_s$ such that Γ pp-defines the following relations*

1. *the relations F_s, $s \in S$,*
2. *the π_s-preimage of the equality relations on E_s, $s \in S$, and*
3. *the π-preimage of every relation in Δ,*

where by the π-preimage of a k-ary relation $Q \subseteq E_{s_1} \times \dots \times E_{s_k}$ over $\mathcal{E}$ we mean the m-ary relation $\pi^{-1}(Q)$ over $\mathcal{D}$, with $m = \sum_{i=1}^{k} \ell_{s_i}$, defined by

$$\pi^{-1}(Q)(x_{1,1}, \dots, x_{1,\ell_{s_1}}, x_{2,1}, \dots, x_{2,\ell_{s_2}}, \dots, x_{k,1}, \dots, x_{k,\ell_{s_k}}) \qquad \textit{is true}$$

if and only if

$$Q(\pi_{s_1}(x_{1,1}, \dots, x_{1,\ell_{s_1}}), \dots, \pi_{s_k}(x_{k,1}, \dots, x_{k,\ell_{s_k}})) \qquad \textit{is true.}$$

▶ **Example 7.** Suppose $\mathcal{D} = \{\mathbb{Z}_2, \mathbb{Z}_3\}$ and $\mathcal{E} = \{\mathbb{Z}_6\}$. Now, any relation on $\mathcal{E}$ is pp-interpretable in a language in $\mathcal{D}$ via $F = \mathbb{Z}_2 \times \mathbb{Z}_3$ and $\pi : F \to \mathbb{Z}_6$ as $\pi(0,0) = 0,\ \pi(1,2) = 1,\ \pi(0,1) = 2,\ \pi(1,0) = 3,\ \pi(0,2) = 4,\ \pi(1,1) = 5$.

As in the one-sorted case, pp-definitions and pp-interpretations give rise to reductions between IMPs. The proof of the following theorem is similar to that of Theorems 3.11 and 3.15 in [15].

▶ **Theorem 8.** *Let Γ, Δ be multi-sorted constraint languages over collections of sets $\mathcal{D} = \{D_t \mid t \in T\}, \mathcal{E} = \{E_s \mid s \in S\}$, respectively.*

(1) *If Γ pp-defines Δ, then IMP(Δ) [IMP$_d(\Delta)$] is polynomial time reducible to IMP(Γ) [respectively, to IMP$_d(\Gamma)$]*

(2) *If Γ pp-interprets Δ, then IMP$_d(\Delta)$ is polynomial time reducible to IMP$_{O(d)}(\Gamma)$.*

3.2 Polymorphisms and multi-sorted polymorphisms

One of the standard methods to reason about constraint satisfaction problems is to use polymorphisms. Here we only give the necessary basic definitions. For more details the reader is referred to [3, 13]. Let R be an (n-ary) relation on a set D and f a (k-ary) operation on the same set, that is, $f : D^k \to D$. Operation f is said to be a *polymorphism* of R, or R is *invariant* under f, if for any $\mathbf{a}_1, \dots, \mathbf{a}_k \in R$ the tuple $f(\mathbf{a}_1, \dots, \mathbf{a}_k)$ belongs to R, where f is applied component-wise, that is,

$$f(\mathbf{a}_1, \dots, \mathbf{a}_k) = (f(a_{1,1}, \dots, a_{1,k}), \dots, f(a_{n,1}, \dots, a_{n,k})),$$

and $\mathbf{a}_i = (a_{1,i}, \dots, a_{n,i})$. The set of all polymorphisms of R is denoted $\mathsf{Pol}(R)$. For a constraint language Γ by $\mathsf{Pol}(\Gamma)$ we denote the set of all operations that are polymorphisms of every relation from Γ.

Polymorphisms provide a link between constraint languages and relations pp-definable in those languages. That is for a constraint language Γ and relation R on set A, the relation R is pp-definable in Γ if and only if $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(R)$ [8, 21].

▶ **Corollary 9** ([25, 15]). *Let Γ, Δ be constraint languages on a set D, Δ finite, and $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$. Then CSP$(\Delta)$ is polynomial time reducible to CSP(Γ), and IMP(Δ) is polynomial time reducible to IMP(Γ).*

We will need a version of polymorphisms adapted to multi-sorted relations. Let $\mathcal{D} = \{D_t \mid t \in T\}$ be a collection of sets. A multi-sorted operation on $\mathcal{D}$ is a *functional symbol* f with associated *arity* k along with an interpretation f^{D_t} of f on every set $D_t \in \mathcal{D}$, which is a k-ary operation on D_t. A multi-sorted operation f is said to be a *(multi-sorted) polymorphism* of a multi-sorted relation $R \subseteq D_{t_1} \times \cdots \times D_{t_n}$, $t_1, \dots, t_n \in T$, if for any $\mathbf{a}_1, \dots, \mathbf{a}_k \in R$ the tuple

$$f(\mathbf{a}_1, \dots, \mathbf{a}_k) = (f^{D_{t_1}}(a_{1,1}, \dots, a_{1,k}), \dots, f^{D_{t_n}}(a_{n,1}, \dots, a_{n,k})) \in R.$$

▶ **Example 10.** Note that for the sake of defining a multi-sorted operation, the collection $\mathcal{D}$ does not have to be finite. Let $\mathcal{A}$ be the class of all finite Abelian groups and f a ternary functional symbol that is interpreted as the affine operation $f^{\mathbb{A}}(x, y, z) = x - y + z$ on every $\mathbb{A} \in \mathcal{A}$, where $+, -$ are operations of $\mathbb{A}$. Consider the multi-sorted binary relation $R \subseteq \mathbb{Z}_2 \times \mathbb{Z}_4$ over $\mathcal{D} = \{\mathbb{Z}_2, \mathbb{Z}_4\}$ given by $R = \{(0,1), (0,3), (1,0), (1,2)\}$. It is straightforward to verify that f is a polymorphism of R. For instance,

$$f\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\2\end{pmatrix}\right) = \begin{pmatrix}0-1+1\\1-0+2\end{pmatrix} = \begin{pmatrix}0\\3\end{pmatrix} \in R.$$

To make sure f is a polymorphism of R we of course have to check every combination of pairs from R.

The connection between multi-sorted polymorphisms and pp-definitions is more complicated than that in the one-sorted case [12], and we do not need it here.

4 CSPs and IMPs over Abelian groups

In this section we outline a proof of our main result, Theorem 11.

▶ **Theorem 11.** *Let $\mathbb{A}$ be an Abelian group. Then IMP$_d(\Delta)$ is polynomial time decidable for any d and any finite constraint language Δ which is invariant under the affine operation of $\mathbb{A}$. Moreover, a proof of membership for IMP$_d(\Delta)$ can also be found in polynomial time.*

Let $\mathbb{A}$ be an Abelian group and Δ a constraint language invariant with respect to the operation $x-y+z$ of $\mathbb{A}$. We first show how a given instance $\mathcal{P}$ of CSP(Δ) can be transformed in such a way that a Gröbner Basis of the resulting instance can be constructed. Then we use substitution reductions to extend this reduction to instances of IMP$_d(\Delta)$.

Step 1: Reduction to a multisorted language over cyclic groups. As was mentioned in the introduction, the standard way to solve CSP(Δ) and IMP$_d(\Delta)$ for languages over $\mathbb{Z}_p$ is to represent instances as a system of linear equations. However, it is not always possible for general Abelian groups. For example, the relation R below over $\mathbb{Z}_2 \times \mathbb{Z}_2$ cannot be represented by a system of linear equations with coefficients from $\mathbb{Z}_2$. This is because there are only 8 linear equations over $\mathbb{Z}_2$ with two variables, and the pairs from R only satisfy the trivial one $0x + 0y = 0$, however, R is nontrivial.

$$R = \left(\begin{array}{cccc} (0,0) & (1,0) & (0,1) & (1,1) \\ (0,0) & (0,1) & (1,0) & (1,1) \end{array} \right) \begin{array}{l} \leftarrow x \\ \leftarrow y \end{array} \tag{1}$$

By the Fundamental Theorem of Abelian groups, $\mathbb{A}$ is a direct sum $\mathbb{Z}_{t_1} \oplus \cdots \oplus \mathbb{Z}_{t_s}$ where each t_i is a prime power and $\mathbb{Z}_{t_i}$ is a cyclic group of order t_i. Using this fact we construct a multisorted constraint language Γ over $\mathbb{Z}_{t_1}, \dots, \mathbb{Z}_{t_s}$ such that Γ pp-interprets Δ and Γ is invariant under the (multisorted) operation $x - y + z$ of $\mathbb{Z}_{t_1}, \dots, \mathbb{Z}_{t_s}$. Moreover, the construction can be amended in such a way that we may assume that t_i, t_j are relatively prime for any $i \neq j$. (However, in this case the direct sum of $\mathbb{Z}_{t_1}, \dots, \mathbb{Z}_{t_s}$ is no longer $\mathbb{A}$.) The following example illustrates the construction.

▶ **Example 12.** Applying such a transformation to the relation R from equation (1) above, every element of $\mathbb{Z}_2 \times \mathbb{Z}_2$ is replaced with a pair of elements of $\mathbb{Z}_2$ in the straightforward way, and R itself is replaced with the 4-ary relation $R' = \{(0,0,0,0), (1,0,0,1), (0,1,1,0), (1,1,1,1)\}$.

By Theorem 8 we have

▶ **Lemma 13.** *For any d the problem IMP$_d(\Delta)$ is polynomial time reducible to IMP$_{O(d)}(\Gamma)$.*

Step 2: Decomposition of multisorted constraints. Fix an instance $\mathcal{P}$ of CSP(Γ). By the following result we may assume that every constraint of Γ is over variables of the same sort.

▶ **Proposition 14.** *Let $\mathcal{P}$ be an instance of CSP(Γ), where Γ is a multi-sorted constraint language over $\mathcal{D} = \{\mathbb{Z}_{t_1}, \dots, \mathbb{Z}_{t_s}\}$ invariant with respect to the affine polymorphism of $\mathbb{Z}_{t_1}, \dots, \mathbb{Z}_{t_s}$, where $t_1, \dots, t_s$ are relatively prime. Then $\mathcal{P}$ is equivalent to $\mathcal{P}'$ such that for every constraint $\langle \mathbf{s}, R \rangle$ of $\mathcal{P}'$, the variables in $\mathbf{s}$ are of the same sort. Moreover, the set of variables X of $\mathcal{P}'$ is the same as that of $\mathcal{P}$ and for any $x \in X$ its sort is the same in both $\mathcal{P}$ and $\mathcal{P}'$.*

Step 3: Constructing a system of linear equations. Step 2 allows us to consider only constraints over $\mathbb{Z}_{p^m}$, p prime. Generally, such relations cannot be represented by a system of linear equations of the form we need, i.e., reduced to a row-echelon form. However, it is possible if new variables are allowed.

▶ **Lemma 15.** *Let R be an n-ary relation invariant under the affine operation of $\mathbb{Z}_{p^m}$. Then there are k and $\alpha_{ij} \in \mathbb{Z}_{p^m}$, $i \in [n], j \in [k]$, such that*

$$R = \{(x_1, \dots, x_n) \mid x_i = \alpha_{i1}y_1 + \dots + \alpha_{ik}y_k, \text{ for } i \in [n], y_1, \dots, y_k \in \mathbb{Z}_{p^m}\}.$$

Lemma 15 allows us to represent an instance of IMP(Γ) as a system of linear equations as follows.

▶ **Proposition 16.** *Every instance $(f_0, \mathcal{P})$ of IMP$_d(\Delta)$ can be transformed to an instance $(f_0', \mathcal{P}')$ of IMP$_{O(d)}(\Gamma)$ satisfying the following conditions and such that $f_0 \in \mathtt{I}(\mathcal{P})$ if and only if $f_0' \in \mathtt{I}(\mathcal{P}')$.*

(1) *For every $i \in [s]$ there is a set $Y_i = \{y_{i,1}, \dots, y_{i,r_i}\}$ of variables of $\mathcal{P}'$ and $Y_i \cap Y_j = \emptyset$ for $i \neq j$.*

(2) *For every constraint $\langle \mathbf{s}, R\rangle$ of $\mathcal{P}'$ the following conditions hold:*

  (a) *there is $i \in [s]$ such that $\mathbb{Z}_{p_i^{m_i}}$ is the domain of every variable from $\mathbf{s}$;*

  (b) *R is represented by a system of linear equations of the form $x_j = \alpha_1 y_{i,1} + \dots + \alpha_{r_i} y_{i,r_i}$, $x_j \in \mathbf{s}$, over $\mathbb{Z}_{p_i^{m_i}}$.*

Let $\mathcal{L}_i$ denote the collection of all equations constructed in Proposition 16 for constraints over $\mathbb{Z}_{p_i^{m_i}}$.

▶ **Example 17.** The relation R' from Example 12 can be represented by the following system of linear equations that uses two extra parameters y_1, y_2:

$$x_1 = y_1, \quad x_2 = y_2, \quad x_3 = y_2, \quad x_4 = y_1.$$

Step 4: Reduction to roots of unity. Using Proposition 16 we can construct a Gröbner Basis of instance $\mathcal{P}$ of CSP(Γ) as follows. Note first of all that a system of linear equations over $\mathbb{Z}_{p_i^{m_i}}$ can be solved in polynomial time. This immediately tells us if $1 \in \mathtt{I}(\mathcal{P})$ or not, and we proceed only if $1 \notin \mathtt{I}(\mathcal{P})$. Let $x_{1,1}, \dots, x_{1,k_1}, \dots, x_{s,1}, \dots, x_{s,k_s}$ and $y_{1,1}, \dots, y_{1,r_1}, \dots, y_{s,1}, \dots, y_{s,r_s}$ be the variables of $\mathcal{P}$ and assume the lexicographic order $\succ_{\mathsf{lex}}$ with

$$\begin{aligned} x_{1,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{1,k_1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{s,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{s,k_s} \\ \succ_{\mathsf{lex}} y_{1,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} y_{1,r_1} \succ_{\mathsf{lex}} y_{2,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} y_{2,r_2} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} y_{s,r_s}. \end{aligned} \tag{2}$$

Since the systems $\mathcal{L}_i$ of linear equations do not share any variables we construct a Gröbner Basis for each of them independently. Then we show that the union of all these Gröbner Bases is indeed a Gröbner Basis for $\mathtt{I}(\mathcal{P})$. For each $\mathcal{L}_i$ we denote the corresponding ideal by $\mathtt{I}(\mathcal{L}_i)$.

Each linear system $\mathcal{L}_i$ is already in its reduced row-echelon form with $x_{i,j}$ as the leading monomial of the j-th equation, $1 \leq j \leq k_i$. Each linear equation can be written as $x_{i,j} + f_{i,j} = 0 \pmod{p_i^{m_i}}$ where $f_{i,j}$ is a linear polynomial over $\mathbb{Z}_{p_i^{m_i}}$. Hence, a generating set for $\mathtt{I}(\mathcal{L}_i)$ in an implicit form is as follows where the addition is modulo $\mathbb{Z}_{p_i^{m_i}}$,

$$G_i = \left\{x_{i,1} + f_{i,1}, \dots, x_{i,k_i} + f_{i,k_i}, \prod_{j \in \mathbb{Z}_{p_i^{m_i}}} (y_{i,1} - j), \dots, \prod_{j \in \mathbb{Z}_{p_i^{m_i}}} (y_{i,r_i} - j)\right\} \tag{3}$$

Unfortunately, a polynomial representation of $x_{i,j} + f_{i,j}$ is exponentially large, and so we need an extra step.

Let $U_{p_i^{m_i}} = \{\omega_i, \omega_i^2, \ldots, \omega_i^{(p_i^{m_i})} = \omega_i^0 = 1\}$ be the set of $p_i^{m_i}$-th roots of unity where ω_i is a primitive $p_i^{m_i}$-th root of unity. For a primitive $p_i^{m_i}$-th root of unity ω_i we have $\omega_i^a = \omega_i^b$ if and only if $a \equiv b \pmod{p_i^{m_i}}$. From $\mathcal{L}_i$ we construct a new CSP instance $\mathcal{L}'_i = (V, U_{p_i^{m_i}}, \widetilde{C})$ as follows. For each equation $x_{i,t} + f_{i,t} = 0 \pmod{p_i^{m_i}}$, where

$$f_{i,t} = \alpha_{i,t,1} y_{i,1} + \cdots + \alpha_{i,t,r_i} y_{i,r_i} + \alpha_{i,t},$$

we add the constraint $x_{i,t} - f'_{i,t} = 0$ (here subtraction is in $\mathbb{C}$) with

$$f'_{i,t} = \omega_i^{\alpha_{i,t}} \cdot \left(y_{i,1}^{\alpha_{i,t,1}} \cdot \cdots \cdot y_{i,r_i}^{\alpha_{i,t,r_i}} \right).$$

Moreover, the domain constraints are different. For each variable $x_{i,j}$, $j \in [k_i]$, or $y_{i,j}$, $j \in [r_i]$ the domain polynomial is $(x_{i,j})^{(p_i^{m_i})} - 1$, $(y_{i,j})^{(p_i^{m_i})} - 1$. However, we do not need the domain polynomials for variables $x_{i,j}$.

▶ **Lemma 18.** *The set of polynomials $G' = \cup_{1 \le i \le s} G'_i$, where*

$$G'_i = \left\{ x_{i,1} - f'_{i,1}, \ldots, x_{i,k_i} - f'_{i,k_i}, (y_{i,1})^{(p_i^{m_i})} - 1, \ldots, (y_{i,r_i})^{(p_i^{m_i})} - 1 \right\}$$

forms a Gröbner Basis for $\mathtt{I}(\mathcal{P}') = \mathbf{I}(\mathsf{Sol}(\mathcal{P}'))$ with respect to the lex *order* (2).

Step 5: Transforming the input polynomial. Given an instance $(f_0, \mathcal{P})$ of $\mathrm{IMP}_d(\Delta)$ Steps 1–4 transform $\mathcal{P}$ to an ideal over the set of roots of unity, for which a Gröbner Basis can be efficiently constructed. To complete a solution algorithm for $\mathrm{IMP}_d(\Delta)$ we need to demonstrate how to convert the input polynomial f_0.

To this end note that the reduction in Step 1 converted f_0 into a polynomial f'_0 over $x_{1,1}, \ldots, x_{1,k_1}, \ldots, x_{s,1}, \ldots, x_{s,k_s}$, see Theorem 8 and Lemma 13. Then for each $i \in [s]$ we define a univariate polynomial $\phi_i \in \mathbb{C}[X]$ that interpolates points $(\omega_i^0, 0), (\omega_i, 1), \ldots,$ $(\omega_i^{(p_i^{m_i}-1)}, p_i^{m_i} - 1)$, that is, $\phi_i(a) = \omega_i^a$ for $a \in \mathbb{Z}_{p_i^{m_i}}$.

▶ **Lemma 19.** *Define polynomial $f''_0 \in \mathbb{C}[X]$ to be*

$$\begin{aligned} & f''_0(x_{1,1}, \ldots, x_{1,k_1}, \ldots, x_{s,1}, \ldots, x_{s,k_s}) \\ & \qquad = f'_0 \left(\phi_1^{-1}(x_{1,1}), \ldots, \phi_1^{-1}(x_{1,k_1}), \ldots, \phi_s^{-1}(x_{s,1}), \ldots, \phi_s^{-1}(x_{s,k_s}) \right). \end{aligned}$$

Then $f_0 \in \mathtt{I}(\mathcal{P})$ if and only if $f''_0 \in \mathtt{I}(\mathcal{P}')$.

If f_0 has degree at most d, the polynomial f''_0 has degree $O(d)$, and thus can be constructed in polynomial time. Therefore, Lemma 19 completes the proof of the first part of Theorem 11. The search version of $\mathrm{IMP}_d(\Delta)$ is discussed in the next section.

5 Search version and the substitution technique

In [15] we introduced a framework to bridge the gap between the decision and the search versions of the IMP. Indeed, this framework gives a polynomial time algorithm to construct a truncated Gröbner Basis provided that the search version of a variation of the IMP is polynomial time solvable. This variation is called χIMP and is defined as follows.

▶ **Definition 20** (χIMP)**.** *Given an ideal* $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ *and a vector of* ℓ *polynomials* $M = (g_1, \ldots, g_\ell)$*, the* χ*IMP asks if there exist coefficients* $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ *such that* $\mathbf{c}M = \sum_{i=1}^{\ell} c_i g_i$ *belongs to the ideal* $\mathtt{I}$*. In the search version of the problem the goal is to find coefficients* $\mathbf{c}$*.*

The χIMP associated with a (multi-sorted) constraint language Γ over a set $\mathcal{D}$ is the problem $\chi\mathrm{IMP}(\Gamma)$ in which the input is a pair $(M, \mathcal{P})$ where $\mathcal{P}$ is a CSP(Γ) instance and M is a vector of ℓ polynomials. The goal is to decide whether there are coefficients $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ such that $\mathbf{c}M$ lies in the combinatorial ideal $\mathcal{I}(\mathcal{P})$. We use $\chi\mathrm{IMP}_d(\Gamma)$ to denote $\chi\mathrm{IMP}(\Gamma)$ when the vector M contains polynomials of degree at most d.

▶ **Theorem 21** (Theorem 1 part (2) paraphrased)**.** *Let* $\mathcal{H}$ *be a class of ideals for which the search version of* χ*IMP*$_d$ *is polynomial time solvable. Then there exists a polynomial time algorithm that constructs a d-truncated Gröbner Basis of an ideal* $\mathtt{I} \in \mathcal{H}$*,* $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$*, in time* $O(n^d)$*.*

The above theorem suggests that, in order to prove the second part of Theorem 11, it is sufficient to show that χIMP is polynomial time solvable for instances of CSP arising from constraint languages that are closed under the affine operation of an Abelian group.

It was shown in [15] that having a Gröbner Basis yields a polynomial time algorithm for solving the search version of χIMP (by using the *division algorithm* and solving a system of linear equations).

▶ **Theorem 22** ([15])**.** *Let* $\mathtt{I}$ *be an ideal, and let* $\{g_1, \ldots, g_s\}$ *be a Gröbner Basis for* $\mathtt{I}$ *with respect to some monomial ordering. Then the (search version of)* χ*IMP is polynomial time solvable.*

Given the above theorem, to solve the χIMP one might reduce the problem at hand to a problem for which a Gröbner Basis can be constructed in a relatively simple way. This approach has been proven to be extremely useful in various cases studied in [15]. In that paper the reductions for χIMP are proved in an ad hoc manner. However, the core idea in all of them is a substitution technique. Here we provide a unifying construction based on *substitution reductions* that covers all the useful cases so far.

5.1 Reduction by substitution

We call a class of χIMPs *CSP-based* if its instances are of the form $(M, \mathcal{P})$, where $\mathcal{P}$ is a CSP instance over a fixed set D. Let $\mathcal{X}, \mathcal{Y}$ be restricted CSP-based classes of the χIMP. The classes $\mathcal{X}, \mathcal{Y}$ can be defined by various kinds of restrictions, for example, as $\chi\mathrm{IMP}(\Gamma), \chi\mathrm{IMP}(\Delta)$, but not necessarily. Let the domain of $\mathcal{X}$ be D and the domain of $\mathcal{Y}$ be E. Let also $\mu_1, \ldots, \mu_k$ be a collection of surjective functions $\mu_i : E^{\ell_i} \to D$, $i \in [k]$. Each mapping μ_i can be interpolated by a polynomial h_i. We call the collection $\{h_1, \ldots, h_k\}$ a *substitution collection.*

The problem $\mathcal{X}$ is said to be *substitution reducible* to $\mathcal{Y}$ if there exists a substitution collection $\{h_1, \ldots, h_k\}$ and a polynomial time algorithm A such that for every instance $(M, \mathcal{P})$ of $\mathcal{X}$ an instance constructed as follows belongs to $\mathcal{Y}$.

(1) Let X be the set of variables of $(M, \mathcal{P})$. For every $x \in X$ the algorithm A selects a polynomial h_{i_x} and a set of variables Y_x such that
  - **(a)** $|Y_x| = \ell_{i_x}$;
  - **(b)** for any $x, y \in X$ either $Y_x = Y_y$ or $Y_x \cap Y_y = \emptyset$;
  - **(c)** if $x_1, \ldots, x_r \in X$ are such that $Y_{x_1} = \cdots = Y_{x_r} = \{y_1, \ldots, y_{\ell_j}\}$ then for any solution φ of $\mathcal{P}$ there are values $a_1, \ldots, a_{\ell_j} \in E$ such that $\varphi(x_i) = h_{i_{x_i}}(a_1, \ldots, a_{\ell_j})$.

(2) If $M = (g_1, \ldots, g_\ell)$ then $M' = (g'_1, \ldots, g'_\ell)$, where for $g_i(x_1, \ldots, x_t)$

$$g'_i = g_i(h_{i_{x_1}}(Y_{x_1}), \ldots, h_{i_{x_y}}(Y_{x_t})).$$

(3) Let $Y = \bigcup_{x \in X} Y_x$. The instance $\mathcal{P}'$ is given by $(Y, E, \mathcal{C}')$, where for every constraint $\langle \mathbf{s}, R \rangle$, $\mathbf{s} = (x_1, \ldots, x_t)$, $\mathcal{P}'$ contains the constraint $\langle \mathbf{s}', R' \rangle$ such that
- $\mathbf{s}' = (x_{1,1}, \ldots, x_{1,\ell_{x_1}}, x_{2,1}, \ldots, x_{t,\ell_{x_t}})$, where $Y_{x_j} = \{x_{j,1}, \ldots, x_{j,\ell_j}\}$;
- R' is an ℓ-ary relation, $\ell = \ell_{x_1}+, \ldots, +\ell_{x_t}$, such that $(a_{1,1}, \ldots, a_{1,\ell_{x_1}}, a_{2,1}, \ldots, a_{t,\ell_{x_t}}) \in R'$ if and only if $(h_{i_{x_1}}(a_{1,1}, \ldots, a_{1,\ell_{x_1}}), \ldots, h_{i_{x_t}}(a_{t,1}, \ldots, a_{t,\ell_{x_t}})) \in R$.

▶ **Lemma 23.** *Let $\mathcal{X}, \mathcal{Y}$ be restricted CSP-based classes of the χIMP$_d$ and χIMP$_{rd}$, respectively, $r \geq 1$. If $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$ with a substitution collection $\{h_1, \ldots, h_k\}$, and $r \geq \ell_i$ for each $i \in [k]$, then there is a polynomial time reduction from $\mathcal{X}$ to $\mathcal{Y}$.*

Since the search χIMP can be solved whenever a Gröbner Basis can be efficiently found, the above lemma provide a powerful tool for solving the χIMP. That is, if $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$ and furthermore $\mathcal{Y}$ is such that it admits a polynomial time algorithm to construct a Gröbner Basis, then instances of $\mathcal{X}$ are solvable in polynomial time too.

▶ **Theorem 24.** *Let $\mathcal{X}, \mathcal{Y}$ be restricted CSP-based classes of the χIMP$_d$ and χIMP$_{rd}$, $r \geq 1$ respectively, such that $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$ with a substitution collection $\{h_1, \ldots, h_k\}$ and $r \geq \ell_i$ for $i \in [k]$. Suppose there exists a polynomial time algorithm that for any instance $(M', \mathcal{P}')$ of $\mathcal{Y}$ constructs a (truncated) Gröbner Basis, then*
1. *there is a polynomial time algorithm that solves every instance $(M, \mathcal{P})$ of $\mathcal{X}$; and*
2. *there is a polynomial time algorithm that for any instance $(M, \mathcal{P})$ of $\mathcal{X}$ constructs a d-truncated Gröbner Basis for $\mathtt{I}(\mathcal{P})$.*

We point out that the second part of Theorem 24 follows from Theorem 21, that is, since every instance $(M, \mathcal{P})$ of $\mathcal{X}$ is polynomial time solvable, by Theorem 21, we can construct a d-truncated Gröbner Basis for $\mathtt{I}(\mathcal{P})$ in polynomial time.

If $\mathcal{X}, \mathcal{Y}$ are of the form χIMP(Γ), Theorem 24 implies the following corollary, which covers virtually all the reductions suggested in [15].

▶ **Corollary 25.** *Let Δ and Γ be multi-sorted constraint languages over finite collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$, $\mathcal{E} = \{E_s \mid s \in S\}$, respectively. Suppose Γ pp-interprets Δ and there exists a polynomial time algorithm that for any instance $(M', \mathcal{P}')$ of χIMP$_{O(d)}(\Gamma)$ constructs a (truncated) Gröbner Basis, then*
1. *there is a polynomial time algorithm that solves every instance $(M, \mathcal{P})$ of χIMP$_d(\Delta)$; and*
2. *there is a polynomial time algorithm that for any instance $(M, \mathcal{P})$ of χIMP$_d(\Delta)$ constructs a d-truncated Gröbner Basis for $\mathtt{I}(\mathcal{P})$.*

Given Corollary 25, we can prove the reductions in Steps 1–5 are reductions by substitution (see the full version [16]), thus by Theorem 24 we can construct a d-truncated Gröbner Basis which yields the search version of Theorem 11.

References

1 Albert Atserias and Joanna Ochremiak. Proof complexity meets algebra. *ACM Trans. Comput. Log.*, 20(1):1:1–1:46, 2019.

2 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, 2014.

3 Libor Barto, Andrei A. Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei A. Krokhin and Stanislav Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

4 Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bound on hilbert's nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, FOCS 1994, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365714.

5 Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem and a majority polymorphism over the ternary domain. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 13:1–13:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.13.

6 Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem for boolean minority. *CoRR*, abs/2006.16422, 2020. arXiv:2006.16422.

7 Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem for boolean minority and dual discriminator. In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*. To appear, 2021.

8 V.G. Bodnarchuk, L.A. Kaluzhnin, V.N. Kotov, and B.A. Romov. Galois theory for post algebras. i. *Kibernetika*, 3:1–10, 1969.

9 Bruno Buchberger. Bruno buchberger's phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3):475–511, 2006. Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday). doi:10.1016/j.jsc.2005.09.007.

10 Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017.

11 Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

12 Andrei A. Bulatov and Peter Jeavons. An algebraic approach to multi-sorted constraints. In *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2003. doi:10.1007/978-3-540-45193-8_13.

13 Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.

14 Andrei A. Bulatov, Andrei A. Krokhin, and Benoît Larose. Dualities for constraint satisfaction problems. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*, pages 93–124. Springer, 2008.

15 Andrei A. Bulatov and Akbar Rafiey. On the complexity of csp-based ideal membership problems. *CoRR*, abs/2011.03700, 2020.

16 Andrei A. Bulatov and Akbar Rafiey. The ideal membership problem and abelian groups. *CoRR*, abs/2201.05218, 2022.

17 Samuel R. Buss and Toniann Pitassi. Good degree bounds on nullstellensatz refutations of the induction principle. In Steven Homer and Jin-Yi Cai, editors, *Proceedings of the 11th Annual IEEE Conference on Computational Complexity, CCC 1996, Philadelphia, Pennsylvania, USA, May 24-27, 1996*, pages 233–242. IEEE Computer Society, 1996. doi:10.1109/CCC.1996.507685.

18 Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In Gary L. Miller, editor, *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, STOC 1996, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 174–183. ACM, 1996. doi:10.1145/237814.237860.

19 David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.

20 Alicia Dickenstein, Noaï Fitchas, Marc Giusti, and Carmen Sessa. The membership problem for unmixed polynomial ideals is solvable in single exponential time. *Discret. Appl. Math.*, 33(1-3):73–94, 1991. doi:10.1016/0166-218X(91)90109-A.

21 D. Geiger. Closed systems of function and predicates. *Pacific Journal of Mathematics*, pages 95–100, 1968.

22 Dima Grigoriev. Tseitin's tautologies and lower bounds for nullstellensatz proofs. In *39th Annual Symposium on Foundations of Computer Science, FOCS 1998, November 8-11, 1998, Palo Alto, California, USA*, pages 648–652. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743515.

23 Grete Hermann. Die frage der endlich vielen schritte in der theorie der polynomideale. *Mathematische Annalen*, 95(1):736–788, 1926.

24 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

25 Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997. doi:10.1145/263867.263489.

26 Christopher Jefferson, Peter Jeavons, Martin J. Green, and M. R. C. van Dongen. Representing and solving finite-domain constraint problems using systems of polynomials. *Annals of Mathematics and Artificial Intelligence*, 67(3):359–382, March 2013. doi:10.1007/s10472-013-9365-7.

27 Monaldo Mastrolilli. The complexity of the ideal membership problem for constrained problems over the boolean domain. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 456–475, 2019. doi:10.1137/1.9781611975482.29.

28 Ernst W. Mayr. Membership in plynomial ideals over Q is exponential space complete. In Burkhard Monien and Robert Cori, editors, *6th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1989, Paderborn, FRG, February 16-18, 1989, Proceedings*, volume 349 of *Lecture Notes in Computer Science*, pages 400–406. Springer, 1989. doi:10.1007/BFb0029002.

29 Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.

30 Ryan O'Donnell. SOS is not obviously automatizable, even approximately. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 59:1–59:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.59.

31 Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 80:1–80:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.80.

32 Fred Richman. Constructive aspects of noetherian rings. *Proceedings of the American Mathematical Society*, 44(2):436–441, 1974.

33 Abraham Seidenberg. Constructions in algebra. *Transactions of the American Mathematical Society*, 197:273–313, 1974.

34 Johan Thapper and Stanislav Zivný. The limits of SDP relaxations for general-valued csps. *ACM Trans. Comput. Theory*, 10(3):12:1–12:22, 2018.

35 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017.

The Aperiodic Domino Problem in Higher Dimension

Antonin Callard
Université Paris-Saclay, ENS Paris-Saclay, Département Informatique,
91190 Gif-sur-Yvette, France

Benjamin Hellouin de Menibus[1]
Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique,
91400 Orsay, France

Abstract

The classical Domino problem asks whether there exists a tiling in which none of the forbidden patterns given as input appear. In this paper, we consider the aperiodic version of the Domino problem: given as input a family of forbidden patterns, does it allow an aperiodic tiling? The input may correspond to a subshift of finite type, a sofic subshift or an effective subshift.

[8] proved that this problem is co-recursively enumerable (Π_0^1-complete) in dimension 2 for geometrical reasons. We show that it is much harder, namely analytic (Σ_1^1-complete), in higher dimension: $d \geq 4$ in the finite type case, $d \geq 3$ for sofic and effective subshifts. The reduction uses a subshift embedding universal computation and two additional dimensions to control periodicity.

This complexity jump is surprising for two reasons: first, it separates 2- and 3-dimensional subshifts, whereas most subshift properties are the same in dimension 2 and higher; second, it is unexpectedly large.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Problems, reductions and completeness

Keywords and phrases Subshift, periodicity, aperiodicity, domino problem, subshift of finite type, sofic subshift, effective subshift, tilings, computability

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.19

Acknowledgements The authors are grateful to the three referees for their many remarks and improvements. The first author is grateful to the *Excellence* group for being such cheerful co-interns and friends.

1 Introduction

Subshifts are sets of colorings (or *configurations*) defined by a family of forbidden patterns. The seminal computational problem on multidimensional subshifts is the *Domino Problem*: given a subshift of finite type (SFT), does it contain a configuration? It was proved undecidable on $\mathbb{Z}^2$ in [2, 19] from the construction of *aperiodic SFTs* (SFTs which contain only (strongly) aperiodic colorings, i.e. colorings with no non-zero period) in which universal computation is embedded. Many similar undecidability results used different SFTs and embeddings to control the structure and properties of their configurations [10, 5, 1, 21, 5] or to characterise the set of possible values of some parameters by computability conditions [12, 15, 4]. These results all rely on the existence of purely aperiodic SFTs on $\mathbb{Z}^d$ for $d \geq 2$ (see [14, Section 1.2] for more details), and show that multidimensional SFTs can be considered as geometrical computational models.

[1] Corresponding author

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 19; pp. 19:1–19:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In contrast, topological or geometrical restrictions may lower the "natural" complexity of a problem (compare e.g. [12] with [6] or [18]) by breaking our ability to embed computation. In particular, finding the border where the difficulty jump occurs gives a fine understanding of the effect of the restriction [7].

Given the importance of aperiodicity for computation embedding, it is natural to ask the aperiodic version of the Domino problem (**AD**): *given as input a subshift X, does it contain an aperiodic coloring?* It is not difficult to see that this problem is harder than the Domino problem, i.e. co-r.e.(Π_1^0)-hard in dimension 2 and higher; however, the natural upper bound is much higher, outside the arithmetical hierarchy.

This question was (to the best of our knowledge) first explored in [8]: the authors proved that **AD** is Π_1^0-complete for $\mathbb{Z}^2$ subshifts[2]. It is an example of problem whose computational complexity is low because of geometrical reasons specific to the two-dimensional case: starting from an aperiodic configuration, we can regroup breaks of periods into concentric balls whose size is controlled by a computable function ([8, Theorem 1]).

In this paper, we study the computational complexity of this problem in higher dimension, where this geometrical property no longer holds (see [8, Section 4] for a counter example). We build an embedding for universal computation that proves that this problem is in a much higher undecidability class – Σ_1^1-complete, its natural upper bound – in sofic subshifts for $d \geq 3$ and in subshifts of finite type for $d \geq 4$.

Our paper is structured as follows.

- In Section 2, we provide definitions for subshifts and the relevant complexity classes;
- In Section 3, we prove that **AD** is Σ_1^1-complete on $\mathbb{Z}^d$ ($d \geq 3$) sofic subshifts;
- In Section 4, we adapt the previous proof to $\mathbb{Z}^d$ ($d \geq 4$) subshifts of finite type;
- In Section 5, we make a side remark relating the existence of an aperiodic configuration in SFTs with their complexity.

We summarize the complexity of **AD** in the following table (new results are highlighted):

Dimension / type	finite type	sofic	effective
2D	Π_1^0-complete	Π_1^0-complete	Π_1^0-complete
3D	open	Σ_1^1-complete	Σ_1^1-complete
4D+	Σ_1^1-complete	Σ_1^1-complete	Σ_1^1-complete

Considering the effect of the dimension on the difficulty of **AD**, we find a border between the dimensions where the complexity of the problem is lowered by geometric properties and the dimensions where computability considerations dominate.

For sofic and effective subshifts, this border lies between dimensions 2 and 3. For SFTs on $\mathbb{Z}^3$, we conjecture that **AD** is in the arithmetical hierarchy for reasons that are specific to SFTs, and provide a few pointers in conclusion. This would be a candidate for a dimension-separating property between 3 and 4 dimensional SFTs. In both cases, we do not know of any other natural problem with a complexity jump in such high dimensions.

2 Definitions and notations

2.1 Subshifts

For a more detailed introduction, we refer the reader to [3, Chapter 9].

[2] on SFTs, but as [8, Theorem 1] applies to any $\mathbb{Z}^2$ subshift, the result also holds for $\mathbb{Z}^2$ effective subshifts.

Let Σ be a finite alphabet of colors and d a dimension. A *configuration* is a coloring $c : \mathbb{Z}^d \mapsto \Sigma$, and the value of c at position i is denoted c_i. A *pattern* is a coloring $w : D \mapsto \Sigma$ of a finite domain $D = \mathrm{dom}(w) \subseteq \mathbb{Z}^2$. We say that a pattern w appears in a configuration x and write $w \sqsubseteq x$ if $w_j = x_{i+j}$ for some $i \in \mathbb{Z}^d$ and all $j \in D$. Given a configuration x and a vector $t \in \mathbb{Z}^d$, denote $\sigma^t(x)$ the shift of x by t: for any $i \in \mathbb{Z}^d, \sigma^t(x)_i = x_{i-t}$.

▶ **Definition 1** (Periodicity).
1. *In a configuration $x \in \Sigma^{\mathbb{Z}^d}$, a vector $p \in \mathbb{Z}^d$ is* broken at position i *if $x_{i+p} \neq x_i$.*
2. *A configuration $x \in \Sigma^{\mathbb{Z}^d}$ is (strongly)* aperiodic *if every vector $p \in \mathbb{Z}^d$ is broken in x.*

In the following definition, Σ is equipped with the discrete topology and $\Sigma^{\mathbb{Z}^d}$ with the product topology. $\Sigma^{\mathbb{Z}^d}$ is then a Cantor space.

▶ **Definition 2** (Subshifts). *A* subshift *is a closed and σ-invariant subset of $\mathbb{Z}^d$. Equivalently, there is a family of forbidden patterns $\mathcal{F}$ such that*

$$X = X_{\mathcal{F}} := \left\{x \in \Sigma^{\mathbb{Z}^d} : \forall w \in \mathcal{F}, w \not\sqsubseteq x\right\}$$

Two distinct families of forbidden patterns may define the same subshift.

▶ **Definition 3** (Classes of subshifts). *A subshift $Y \subseteq \Sigma^{\mathbb{Z}^d}$ is:*
1. of finite type *(*SFT*) if it can be defined by a finite family of forbidden patterns.*
2. sofic *if there exists an SFT $X \subseteq \Sigma'^{\mathbb{Z}^d}$ and a projection $\pi : \Sigma' \mapsto \Sigma$ such that $Y = \pi(X)$.*
3. effective *if it can be defined by a recursively enumerable family of forbidden patterns.*

SFTs are of course sofic and sofic subshifts are effective. On the other direction, effective subshifts are projections of higher-dimensional sofic subshifts; this is a consequence of [10], later improved in the subshift case in [5, 1]. More precisely, $X^{\uparrow} \subseteq \Sigma^{\mathbb{Z}^{d+k}}$ is a $(d+k)$-dimensional *lift* of a subshift $X \subseteq \Sigma^{\mathbb{Z}^d}$ if its configurations are configurations of X repeated along the k additional dimensions. Then:

▶ **Theorem 4** ([5, 1]). *For any $\mathbb{Z}^d$ effective subshift X, its $(d+1)$-dimensional lifts are sofic.*

2.2 Hierarchy of undecidability

Many-one reductions define a preorder on decision problems ("P_1 is easier than P_2"), so we can define hierarchies according to "how far" a problem is from being computable.

Arithmetical hierarchy

Starting from recursively enumerable (Σ_1^0) and co-recursively enumerable (Π_1^0) problems, the arithmetical hierarchy progressively defines higher levels of undecidability.

▶ **Definition 5** (Arithmetical hierarchy). *For a decision problem $P : \mathbb{N} \mapsto \{0,1\}$ and $m \geq 1$,*
1. *$P \in \Sigma_m^0$ if there is a computable relation $R(n, k_1, ..., k_m)$ such that*

$$P(n) = 1 \Leftrightarrow \exists k_1, \forall k_2, \exists k_3, \dots R(n, k_1, \dots, k_m).$$

2. *$P \in \Pi_m^0$ if this definition holds when swapping $\forall$ and $\exists$ quantifiers.*

P is arithmetical *if it belongs to a level of this hierarchy.*

As $\Sigma_m^0 \cup \Pi_m^0 \subsetneq \Sigma_{m+1}^0 \cap \Pi_{m+1}^0$, this indeed defines a hierarchy. For more details, we refer the reader to [20, Chapter 4].

Analytical hierarchy

Above the arithmetical hierarchy, the analytical hierarchy allows for second-order quantifications on sets. Here we need only the first level.

▶ **Definition 6** (Class Σ^1_1). *A decision problem* $P : \mathbb{N} \mapsto \{0,1\}$ *is* Σ^1_1 *if there exists an* arithmetical *relation* R *such that*

$$P(n) = 1 \Leftrightarrow \exists f \in 2^{\mathbb{N}}, R^f(n)$$

in which R^f *denotes the relation* R *with* f *given as an oracle.*

All arithmetical sets are Σ^1_1. In terms of computational power, Σ^1_1 sets are (a lot) harder than arithmetical sets: to make an analogy between computability and topology, if Σ^0_1 sets correspond to the open sets, then Σ^1_1 sets are not even Borel. For more details, see [17, Chapter IV.2]. A typical example of a Σ^1_1-*complete problem* is the following:

▶ **Theorem 7** (State Recurrence [9, Corollary 6.2]). *The problem of* ***State Recurrence (SR):***
Input: *A nondeterministic Turing machine (NTM)* $\mathcal{M}$, *and one of its states* q_0.
Output: *Is there a run of* $\mathcal{M}$ *on the empty input* ε *in which* q_0 *is visited infinitely often?*
is a Σ^1_1*-complete problem.*

2.3 The aperiodic Domino (AD) problem and its complexity

▶ **Definition 8** (Aperiodic Domino problem (**AD**)).
Input: *An effective family of d-dimensional patterns.*
Output: *Is there an aperiodic configuration in the effective subshift* $X_{\mathcal{F}}$*?*
We consider variations of **AD** depending on the type of input subshift (SFT, sofic, effective). There are natural lower and upper bounds on the complexity of **AD** that do not depend on the input type:

▶ **Proposition 9.** ***AD*** *is* Π^0_1*-hard for* $\mathbb{Z}^d$ *subshifts* ($d \geq 2$).

Proof. We reduce the Domino problem to **AD**. Let Y be a $\mathbb{Z}^d$-SFT with only aperiodic configurations (see e.g. [19]). For any $\mathbb{Z}^d$ subshift X, the cartesian product $X \times Y$ has the same type (SFT, sofic, effective), has only aperiodic configurations, and is non-empty if and only if X is non-empty. ◀

▶ **Proposition 10.** ***AD*** *is a* Σ^1_1 *problem for* $\mathbb{Z}^d$ *subshifts.*

Proof. Let $\mathcal{F}$ be the effective family of forbidden patterns given as input. The existence of an aperiodic configuration can be written as:

$$\exists x \in \Sigma^{\mathbb{Z}^d}, x \in X_{\mathcal{F}} \text{ and } x \text{ is aperiodic.}$$

Taking any computable encoding between $\Sigma^{\mathbb{Z}^d}$ and $2^{\mathbb{N}}$, we can see that the first (existential) quantifier is of second order and can be written as a quantifier on $2^{\mathbb{N}}$. The rest of the expression is a Π^0_2 relation, and in particular arithmetical (x being given as oracle):

- $x \in X_{\mathcal{F}} \Leftrightarrow \forall w \in \mathcal{F}, \forall i \in \mathbb{Z}^d, x_{|i+\mathrm{dom}(w)} \neq w$;
- x is aperiodic $\Leftrightarrow \forall p \in \mathbb{Z}^d, \exists i \in \mathbb{Z}^d, x_i \neq x_{i+p}$. ◀

3 Σ^1_1-completeness for $\mathbb{Z}^d$ sofic and effective subshifts, $d \geq 3$

▶ **Theorem 11.** *$\mathbf{AD}$ for $\mathbb{Z}^d$ sofic subshifts, $d \geq 3$, is a Σ^1_1-complete problem.*

By Proposition 10, $\mathbf{AD} \in \Sigma^1_1$. We prove Σ^1_1-hardness for $d = 3$ and the higher-dimensional cases will follow.

To prove Σ^1_1-hardness, we reduce (many-one reduction) the problem $\mathbf{SR}$. Let $\mathcal{M}$ be some nondeterministic Turing machine (NTM) and q_0 one of its states. We create a sofic subshift Y_3 which contains an aperiodic configuration if and only if $\mathcal{M}$ admits a run from the empty word which visits q_0 infinitely often. The proof is divided in three parts:

1. Section 3.1: creation of an auxiliary $\mathbb{Z}$ Toeplitz subshift $T_{\mathcal{M}}$;
2. Section 3.2: creation of Y_3 and proof that it is sofic;
3. Section 3.3: proof that $Y_3 \in \mathbf{AD}$ iff $(\mathcal{M}, q_0) \in \mathbf{SR}$.

3.1 $T_{\mathcal{M}}$: $\mathbb{Z}$ Toeplitz corresponding to state sequences of $\mathcal{M}$

In this section, we transform the set of sequences of states in all the runs of $\mathcal{M}$ into a $\mathbb{Z}$ subshift with a convenient structure called *Toeplitz*.

$\mathbb{Z}$ Binary Toeplitz subshift X_T

Consider the substitution σ on the alphabet $\{\rightarrow, \leftarrow, \rightarrowtriangle, \leftarrowtriangle\}$:

$$\sigma(\rightarrow) = \rightarrowtriangle\leftarrow \qquad \sigma(\leftarrow) = \leftarrowtriangle\leftarrow \qquad \sigma(\rightarrowtriangle) = \rightarrowtriangle\rightarrow \qquad \sigma(\leftarrowtriangle) = \leftarrowtriangle\rightarrow$$

Define the $\mathbb{Z}$-subshift X_σ by forbidding every pattern that does not appear in $\sigma^\omega(\rightarrowtriangle)$ (any other seed symbol would yield the same subshift).

$$\sigma^\omega(\rightarrowtriangle) = \rightarrowtriangle\rightarrow\rightarrowtriangle\leftarrow\rightarrowtriangle\rightarrow\leftarrowtriangle\leftarrow\rightarrowtriangle\rightarrow\rightarrowtriangle\leftarrow\leftarrowtriangle\rightarrow\leftarrowtriangle\leftarrow\ \ldots$$

▶ **Definition 12** (Binary Toeplitz subshift). *The* binary Toeplitz subshift X_T *is the image of X_σ under the projection that maps $\{\rightarrow, \rightarrowtriangle\}$ to $\rightarrowtriangle$ and $\{\leftarrow, \leftarrowtriangle\}$ to $\leftarrowtriangle$.*

X_T is a Toeplitz subshift [13]. It corresponds to the "period-doubling" or "ruler" (modulo 2) sequences (resp. A001511 and A096268 in the OEIS). In a configuration of X_T,

Level 1 One position out of two has an alternating sequence of $\rightarrowtriangle$ and $\leftarrowtriangle$;

Level 2 One position out of two *in the remaining positions* (i.e. one out of four) has an alternating sequence of $\rightarrowtriangle$ and $\leftarrowtriangle$, etc.

More generally, a position i is of level ℓ if it has minimal period $2^{\ell+1}$ (cells at positions $i + k2^{\ell+1}$ all have the same value). In a configuration of X_T, there may exist at most one position i which does not have a level (i.e. it is not $2^{\ell+1}$ periodic for any ℓ): we say that $\text{level}_z(i) = \infty$. Given as input a finite pattern of size between 2^n and 2^{n+1}, one can compute all levels $\ell \leq n - 1$.

$T_{\mathcal{M}}$: $\mathbb{Z}$-Toeplitzification of sequences of states of $\mathcal{M}$

▶ **Definition 13** (Toeplitzification of a set of sequences). *Given a set of sequences $A \subseteq \Sigma^{\mathbb{N}}$, we define the corresponding Toeplitzified subshift T_A on the alphabet $\Sigma \times \{\rightarrowtriangle, \leftarrowtriangle\}$ as:*

$$T_A = \left\{ (x, z) \in (\Sigma \times \{\rightarrowtriangle, \leftarrowtriangle\})^{\mathbb{Z}} : \begin{array}{l} z \in X_T, \exists (a_n)_{n \in \mathbb{N}} \in A, \\ \forall i \in \mathbb{Z}, \text{level}_z(i) = \ell \in \mathbb{N} \implies x_i = a_\ell \end{array} \right\}.$$

Note that a position of infinite level may be marked with any symbol of Σ. We cannot force this symbol without breaking the next lemma.

Now take $\Sigma = Q$, the set of states of $\mathcal{M}$, and define $S_{\mathcal{M}}$ as the set of sequences $(s_t)_{t\in\mathbb{N}}$ on the alphabet Σ such that there exists a non-terminating run of $\mathcal{M}$ from the empty input whose state at time t is s_t. Let $T_{\mathcal{M}} := T_{S_{\mathcal{M}}}$ be its Toeplitzification.

▶ **Lemma 14.** *$T_{\mathcal{M}}$ is a $\mathbb{Z}$ effective subshift.*

Proof. This stems from the fact that the set of prefixes of $S_{\mathcal{M}}$ is computable: for any $n \geq 0$, we can enumerate all oracles of non-determinism of $\mathcal{M}$ of size $\leq n$ and compute S_n, the set of finite prefixes of length n in $S_{\mathcal{M}}$.

Consider the following algorithm that defines a family of forbidden patterns. For all n:

- Compute the globally admissible patterns of X_T of size $2^n + 1$; (*Note that the language of patterns of X_T is computable: it is both recursively and co-recursively enumerable.)*
- Compute $\mathcal{S}_n$;
- Forbid all patterns $(u, v) \in (\Sigma \times \{\rightarrow, \leftarrow\})^{2^n+1}$, except if v is a pattern in X_T and there exists a prefix $(s_t)_{0\leq t\leq n} \in \mathcal{S}_n$ such that:

$$\forall i, j \in \mathbb{Z},\ (\mathrm{level}_v(i) = \mathrm{level}_v(j) \leq n) \implies u_i = u_j = s_{\mathrm{level}_v(i)}.$$

This procedure defines an effective subshift E. We prove $E = T_{\mathcal{M}}$. Indeed:

$E \subseteq T_{\mathcal{M}}$ Take $(u, v) \in E$ and $(u^n, v^n) = (x, z)[-2^n, 2^n]$. By definition of E, there exists a finite prefix $s^n \in S_n$ such that for any positions i, j with $\mathrm{level}_{v^n}(i) = \mathrm{level}_{v^n}(j) = \ell$, we have $u^n_i = u^n_j = s_\ell$. This sequence of prefixes is increasing, so it converges towards some sequence $s \in S_{\mathcal{M}}$. Then for any $i, j \in \mathbb{Z}$ such that $\mathrm{level}_v(i) = \mathrm{level}_v(j) = \ell < +\infty$, we have $x_i = x_j = s_\ell$. So $(x, z) \in T_{\mathcal{M}}$.

$T_{\mathcal{M}} \subseteq E$ No pattern forbidden in the algorithm appears in any configuration of $T_{\mathcal{M}}$. ◀

3.2 Y_3: the desired $\mathbb{Z}^3$ subshift

We create a subshift Y_3 which contains an aperiodic configuration if and only if there exists a run of $\mathcal{M}$ on the empty word which visits q_0 infinitely often. As one might expect, each configuration of Y_3 contains the lift of a configuration of $T_{\mathcal{M}}$ corresponding to a run of $\mathcal{M}$. We then add lines to make it aperiodic if and only if q_0 appears infinitely often.

However, every decision of breaking periods must occur locally at every level, without the ability to know whether the future number of visits of q_0 is finite or infinite. Otherwise compactness would create issues: as visits of q_0 can occur arbitrarily late, a position of finite level could be tricked to "believe" that q_0 is visited infinitely often in the future. That is why we will break periods whose size depend on the level of the positions in the Toeplitz structure.

Effective 2D subshifts: $Y_2^{\rightarrow}$ and $Y_2^{\leftarrow}$

A configuration of $Y_2^{\rightarrow}$ is composed of three layers:

- Layer 1 & 2 : it contains a $\mathbb{Z}^2$ lift of a configuration $x' \in T_{\mathcal{M}}$. That is, $\forall i, j : x_{i,j} = x'_i$.
- Layer 3: on the alphabet $\{■, □\}$. For every ℓ and in every column of level ℓ containing $(q_0, \rightarrow)$ on Layers 1 and 2, Layer 3 contains regularly placed ■ cells separated by $2^\ell - 1$ □ cells. Every other cell contains □ on Layer 3.

Formally, $Y_2^{\rightarrow}$ can be written as:

$$\left\{\begin{array}{l} x \in (\Sigma \times \{\rightarrow, \leftarrow\} \times \{\blacksquare, \square\})^{\mathbb{Z}^2} : \exists x' \in T_{\mathcal{M}}, \pi_{1,2}(x) \text{ is a } \mathbb{Z}^2 \text{ lift of } x', \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \exists z \in X_T, \pi_2(x) \text{ is a } \mathbb{Z}^2 \text{ lift of } z, \\ \forall i, j \in \mathbb{Z}, \\ \qquad x_{i,j} = (\cdot, \cdot, \blacksquare) \implies \forall j', x_{i,j'} = (q_0, \rightarrow, \cdot) \\ \qquad x_{i,j} = (q_0, \rightarrow, \cdot) \text{ and } \mathrm{level}_z(i) = \ell \in \mathbb{N} \implies \exists j', \forall j'', x_{i,j''} = (\cdot, \cdot, \blacksquare) \iff 2^\ell \mid (j'' - j') \\ \qquad x_{i,j} = (q_0, \rightarrow, \cdot) \text{ and } \mathrm{level}_z(i) = \infty \implies |\{j' : x_{i,j'} = (\cdot, \cdot, \blacksquare)\}| \leq 1 \end{array}\right\}.$$

Its counterpart $Y_2^{\leftarrow}$ is defined similarly by replacing $\rightarrow$ by $\leftarrow$ in the previous definition. It is clear that both $Y_2^{\rightarrow}$ and $Y_2^{\leftarrow}$ are effective $\mathbb{Z}^2$ subshifts.

Issues with the position of infinite level

Note that $\blacksquare$ symbols break increasingly large periods as levels in the Toeplitz structure increase: by compactness, a position of infinite level can break periods of every size by itself.

This explains why this construction requires two additional dimensions to $T_{\mathcal{M}}$ instead of one: each position in Y_3 will be periodic in one dimension, and breaks periods in the other. This way, the single position of infinite level may break horizontal or vertical periods, but not both.

Sofic 3D subshifts: $Y_3^{\rightarrow}$ and $Y_3^{\leftarrow}$

By Theorem 4, every d-dimensional effective subshift can be lifted into a $d+1$-dimensional sofic subshift. Using this result, we lift $Y_2^{\rightarrow}$ and $Y_2^{\leftarrow}$ into 3D sofic subshifts $Y_3^{\rightarrow}$ and $Y_3^{\leftarrow}$:

$$Y_3^{\rightarrow} = \{x \in (\Sigma \times \{\rightarrow, \leftarrow\} \times \{\blacksquare, \square\})^{\mathbb{Z}^3} : \exists x' \in Y_2^{\rightarrow}, \forall i, k \in \mathbb{Z}, \forall j' \in \mathbb{Z}, x_{i,j',k} = x'_{i,k}\}$$
$$Y_3^{\leftarrow} = \{x \in (\Sigma \times \{\rightarrow, \leftarrow\} \times \{\blacksquare, \square\})^{\mathbb{Z}^3} : \exists x' \in Y_2^{\leftarrow}, \forall i, j \in \mathbb{Z}, \forall k' \in \mathbb{Z}, x_{i,j,k'} = x'_{i,j}\}.$$

Note that the lifts are not made along the same coordinates: a position with $\blacksquare$ *in* $Y_2^{\rightarrow}$ *lifts into a line directed by* $(0,1,0)$ *in* $Y_3^{\rightarrow}$*, and a position with* $\blacksquare$ *in* $Y_2^{\leftarrow}$ *lifts into a line directed by* $(0,0,1)$ *in* $Y_3^{\leftarrow}$*.*

Sofic 3D subshift: Y_3

We obtain Y_3 by "fusing" the two previous subshifts. Formally,

$$Y_3 = \{x \in (\Sigma \times \{\rightarrow, \leftarrow\} \times \{\blacksquare, \square\} \times \{\blacksquare, \square\})^{\mathbb{Z}^3} : \pi_{1,2,3}(x) \in Y_3^{\rightarrow} \text{ and } \pi_{1,2,4}(x) \in Y_3^{\leftarrow}\}.$$

Since $Y_3^{\rightarrow}$ and $Y_3^{\leftarrow}$ are sofic, their cartesian product $Y_3^{\rightarrow} \times Y_3^{\leftarrow}$ is also sofic. Y_3 is the projection on Layers $1, 2, 3, 6$ of $Y_3^{\rightarrow} \times Y_3^{\leftarrow}$ with the additional local condition that the first two layers coincide (i.e. $\pi_{1,2}(x) = \pi_{4,5}(x)$), so it is sofic as well.

▷ Claim 15. A configuration of Y_3:

1. breaks every periodicity vector $(n, \cdot, \cdot)$ for $n \geq 1$.
2. every slice $(i, \cdot, \cdot)$ containing $(q_0, \rightarrow)$ on the first two layers and corresponding to the lift of a single position of level $\ell \in [0, +\infty]$ in $T_{\mathcal{M}}$, is periodic with periods $(0,1,0)$ and $(0,0,2^\ell)$ but breaks every period $(\cdot, \cdot, n)$ for $1 \leq n < 2^\ell$. The same is true with $(q_0, \leftarrow)$ with vectors $(0, 2^\ell, 0)$ and $(0,0,1)$.

Proof.

1. the Toeplitzification of alternating $\rightarrow$ and $\leftarrow$ is aperiodic, so Layer 2 breaks all vectors $(n, \cdot, \cdot)$ for $n \geq 1$.

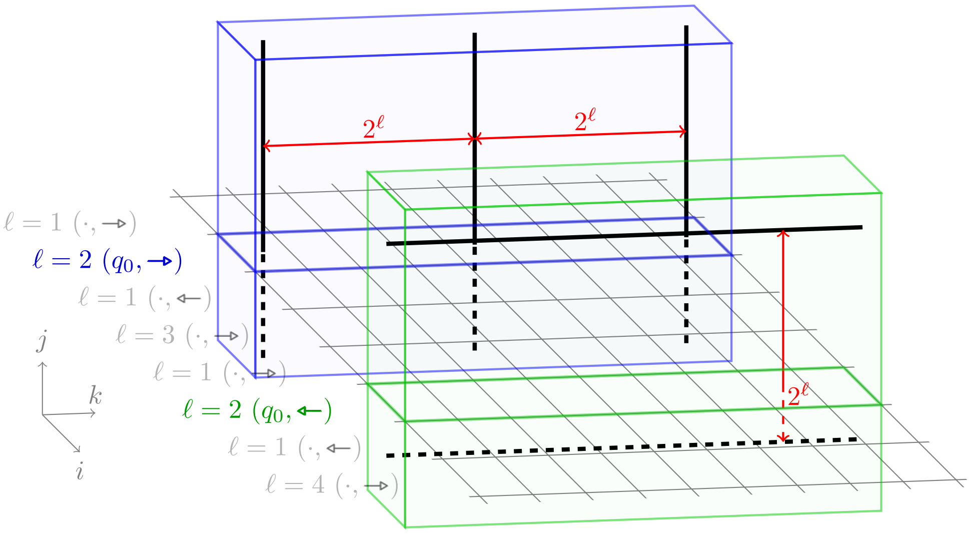

$\ell = 1$ $(\cdot, \rightarrow)$
$\ell = 2$ $(q_0, \rightarrow)$
$\ell = 1$ $(\cdot, \leftarrow)$
$\ell = 3$ $(\cdot, \rightarrow)$
$\ell = 1$ $(\cdot, \rightarrow)$
$\ell = 2$ $(q_0, \leftarrow)$
$\ell = 1$ $(\cdot, \leftarrow)$
$\ell = 4$ $(\cdot, \rightarrow)$
2^ℓ
2^ℓ
2^ℓ
j
k
i

Figure 1 A configuration of Y_3. To the left of each slice $(i, \cdot, \cdot)$ is its level ℓ and the values on Layers 1 and 2. We highlight two slices of level 2: at the front, marked by $(q_0, \leftarrow)$ with horizontal lines; at the back, marked by $(q_0, \rightarrow)$ with vertical lines.

2. Layer 1 and 2 are lifted along the last two dimensions, so they cannot break any such vectors. Layer 3 is lifted along the second dimension so it is $(0, 1, 0)$-periodic, and breaks the required vectors from the last condition in the definition of $Y_2^{\rightarrow}$. Layer 4 is $\square$ everywhere since it is not marked by $(q_0, \leftarrow)$. ◁

3.3 Proof of the reduction RS $\leq$ AD

▶ **Lemma 16.** *A configuration in Y_3 is aperiodic if, and only if, it corresponds to a run of $\mathcal{M}$ in which q_0 occurs infinitely often.*

Proof. Using Claim 15,

- Let $y \in Y_3$ be a configuration corresponding to a run of $\mathcal{M}$ that visit q_0 infinitely often.
  - If $(q_0, \rightarrow)$ appears at a level ℓ, all vectors $(0, \cdot, n)$ for $1 \leq n < 2^\ell$ are broken on Layer 3;
  - Similarly for $(q_0, \leftarrow)$ and vectors $(0, n, \cdot)$ on Layer 4.

  Therefore all vectors $(0, \cdot, \cdot)$ are broken at some level, and vectors $(n, \cdot, \cdot)$ are always broken for $n \geq 1$, so y is an aperiodic configuration.
- Let $y \in Y_3$ be a configuration corresponding to a run of $\mathcal{M}$ that does not visit q_0 after some time $N \in \mathbb{N}$. Let $a_\infty \in \Sigma \times \{\rightarrow, \leftarrow\}$ be the value on Layers 1 and 2 of the *single* position of infinite level in z, if it exists.
  - If $a_\infty \neq (q_0, \leftarrow)$, positions marked by $(q_0, \leftarrow)$ must be of level $\leq N$, so y is periodic of period $(0, 2^N, 0)$.
  - Similarly, if $a_\infty \neq (q_0, \rightarrow)$, then y is periodic of period $(0, 0, 2^N)$.

  All in all, y is not aperiodic. ◀

Case $d > 3$. We lift the previous construction and fill the additional dimensions with aperiodicity. More precisely, in the construction of Y_3, one of the dimension is always aperiodic, and the two others may or may not be periodic. Let A be the $\mathbb{Z}^d$ lift of any $\mathbb{Z}^{d-2}$ aperiodic sofic subshift ($d - 2 \geq 2$), and Y_d the $\mathbb{Z}^d$ lift of Y_3. The cartesian product $A \times Y_d$ is aperiodic if and only if Y_3 is. ◀

4 Σ^1_1-completeness for $\mathbb{Z}^d$ SFTs, $d \geq 4$

▶ **Theorem 17.** *If $d \geq 4$, **AD** for $\mathbb{Z}^d$ SFTs is a Σ^1_1-complete problem.*

As above, we prove Σ^1_1-hardness for $d = 4$, and the result extends to $d > 4$.

4.1 Outline of the proof

This proof has the same structure as Theorem 11 with some adaptations for $\mathbb{Z}^4$ SFTs. We reduce to the problem **SR**: given $\mathcal{M}$ and q_0, we create an SFT X_4 that contains an aperiodic configuration if and only if $\mathcal{M}$ admits a run from the empty word which visits q_0 infinitely often. To do this, we use repeated lines along two dimensions (3 and 4) to break all periods up to a length controlled by a computation embedded in the configuration.

1. In Section 4.2, we build X^2_T, a $\mathbb{Z}^2$ version of the Toeplitz structure X_T;
2. In Section 4.3, we build auxiliary SFTs $X_3^{\rightarrow}$ and $X_3^{\leftarrow}$ (counterparts to $Y_2^{\rightarrow}$ and $Y_2^{\leftarrow}$);
3. In Section 4.4, we build X_4 and prove the reduction.

The main difference is that the finite type case requires an additional dimension to embed computations and some construction lines (dimensions 1 and 2). Remember that the subshift $T_{\mathcal{M}}$ of $\mathbb{Z}$ Toeplitzified sequences of states of runs of $\mathcal{M}$ is effective ; instead, we define an aperiodic $\mathbb{Z}^2$ version $T^2_{\mathcal{M}}$ that is sofic and aperiodic. Since $T^2_{\mathcal{M}}$ is the projection of an aperiodic $\mathbb{Z}^2$ SFT, we then add, as in the previous case, two additional dimensions in which this SFT can be periodic or aperiodic, since the position of infinite level can break periods uncontrollably along at most one dimension.

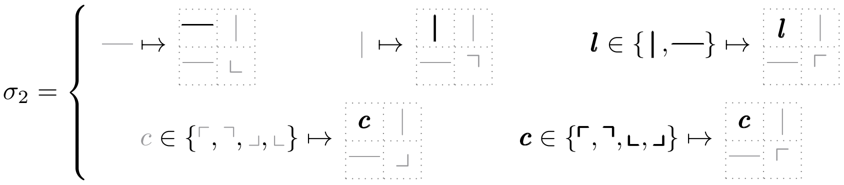
Furthermore, to control the length of the vectors being broken, we need to measure distances between lines (as in Y_3). With SFTs, copying a distance from one dimension to another can only be done with diagonals. Therefore, instead of *lines* to break periods, we use a more complex *diagonal SFT D* that we embed in slices $(i, \cdot, \cdot, \cdot)$ only on dimensions 2 and 3 (on symbols $\rightarrow$) or 2 and 4 (on symbols $\leftarrow$). This way, the computation embedded in the first two dimensions can control the length of the broken periodicity vectors.

4.2 $T^2_{\mathcal{M}}$: $\mathbb{Z}^2$ Toeplitz corresponding to state sequences of $\mathcal{M}$

Binary Toeplitz structure

In this section, we use a $\mathbb{Z}^2$ subshift X^2_T on the alphabet {⌜, ⌝, ⌞, ⌟, |, —} whose structure is a two-dimensional analog of X_T.

It is defined by the substitution σ_2 on the alphabet {⌜, ⌝, ⌞, ⌟, |, —, **⌜**, **⌝**, **⌞**, **⌟**, **|**, **—**}:



As before, X_{σ_2} is the subshift whose forbidden patterns are all the patterns which do not appear in the configuration σ_2^{ω}(**⌜**) (any other seed symbol would do).

▶ **Definition 18** (binary bi-Toeplitz structure)**.** *X^2_T is the color-forgetting projection of X_{σ_2} on the alphabet* {⌜, ⌝, ⌞, ⌟, |, —}.

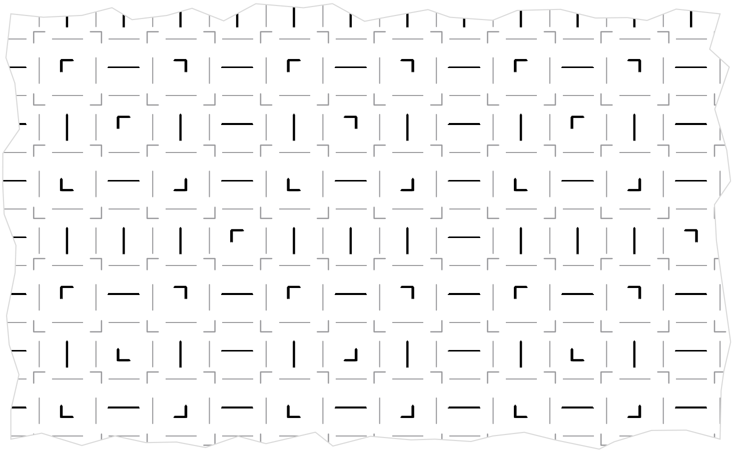


Figure 2 A configuration of X_{σ_2}.

As the substitution σ_2 is deterministic, X^2_T is a sofic subshift by [16, Theorem 4.1].

In a configuration of X^2_T, ignoring symbols $|$ and $—$:

1. corner symbols $\{\ulcorner, \urcorner, \llcorner, \lrcorner\}$ can be grouped together to form squares. A square is of level ℓ if its edges have length 2^ℓ. There may exist a single corner in an otherwise blank line or column: we say it is part of a square of infinite level;
2. each line only contains symbols in either $\{\ulcorner, \urcorner\}$ or $\{\llcorner, \lrcorner\}$, all of the same level. If a line does not contain any corner, its level is said to be infinite;
3. the vertical distance between two consecutive lines of the same level ℓ is 2^ℓ, and those lines contain the same symbols.

The corresponding statements hold for columns.

$T^2_{\mathcal{M}}$: $\mathbb{Z}^2$ Toeplitzification of sequences of states of $\mathcal{M}$

▶ **Definition 19** ($\mathbb{Z}^2$ Toeplitzification of a set of sequences). *Given a set of sequences $A \subseteq \Sigma^{\mathbb{N}}$, we define the corresponding $\mathbb{Z}^2$ Toeplitzified subshift T^2_A on the alphabet $\Sigma_T = \Sigma \times \{\rightarrow, \leftarrow\} \times \{\ulcorner, \llcorner, \urcorner, \lrcorner, |, —\}$ as:*

$$T^2_A = \left\{ x \in (\Sigma_T)^{\mathbb{Z}^2} : \begin{array}{c} \pi_{1,2}(x) \in (T_A)^{\uparrow}, \pi_3(x) \in X^2_T \\ \forall i,j \in \mathbb{Z}, \begin{array}{l} \pi_3(x_{i,j}) \in \{\ulcorner, \llcorner\} \implies \pi_2(x_{i,j}) = \rightarrow \\ \pi_3(x_{i,j}) \in \{\urcorner, \lrcorner\} \implies \pi_2(x_{i,j}) = \leftarrow \end{array} \end{array} \right\}$$

In other words, T^2_A superimposes the structure of a $\mathbb{Z}$ Toeplitzification with the $\mathbb{Z}^2$ structure X^2_T we define above. As before, we denote $T^2_{\mathcal{M}} := T^2_{S_{\mathcal{M}}}$ where $S_{\mathcal{M}}$ of the set of sequences of states corresponding to runs of $\mathcal{M}$.

▶ **Lemma 20.** *$T^2_{\mathcal{M}}$ is a non-empty sofic subshift.*

Proof. $T^2_{\mathcal{M}}$ is non-empty as arrows in $T_{\mathcal{M}}$ can be aligned with corners in X^2_T: arrows of level ℓ as well as columns of level ℓ have period 2^ℓ. For soficness:

- Layers 1 and 2 are $\mathbb{Z}^2$ lifts of configurations of $T_{\mathcal{M}}$, which is an effective subshift (Lemma 14). By Theorem 4, Layers 1 and 2 form a sofic subshift;
- Layer 3 is composed of configurations of X^2_T, which is a sofic subshift;
- the additional condition defining $T^2_{\mathcal{M}}$ (synchronizing Layers 2 and 3) is of finite type. ◀

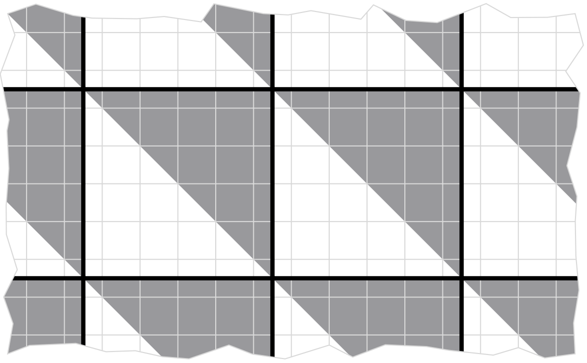
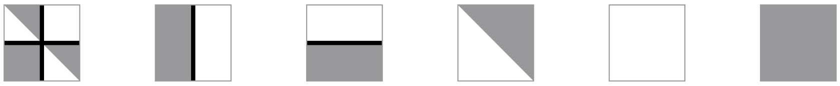


Figure 3 A configuration of D.

4.3 Auxiliary $\mathbb{Z}^2$ and $\mathbb{Z}^3$ SFTs

The diagonal SFT D

The diagonal SFT D is defined by adjacent matching patterns on the alphabet Σ_D:



A configuration of D containing two parallel black lines is in fact periodic and consists of repeated squares. D also contains configurations with 0 or 1 black line.

$\mathbb{Z}^3$ SFTs $X_3^{\rightarrow}$ and $X_3^{\leftarrow}$

This section is written for $X_3^{\rightarrow}$; it applies to $X_3^{\leftarrow}$ by flipping the arrows and corners. As $T^2_{\mathcal{M}}$ is a 3-layer sofic subshift, it is the projection of some 4-layer $\mathbb{Z}^2$ SFT X_2. To create $X_3^{\rightarrow}$, we lift X_2 then add an additional layer for D:

- Layers 1 to 4: $\mathbb{Z}^3$ lift of X_2. In other words, $\pi_{1,2,3}(X_2)$ is a $\mathbb{Z}^3$ lift of $T^2_{\mathcal{M}}$;
- Layer 5 : each slice $(i,\cdot,\cdot)$ contains a configuration $d^i \in D$. If the slice has $(q_0, \rightarrow)$ on its first two layers, then the configuration d^i is "synchronized" with the underlying configuration of $T^2_{\mathcal{M}}$ on Layer 3, that is: vertical lines ┃ in d^i only appear on Layer 5 at positions marked by corners $\ulcorner, \llcorner$ on Layer 3. Otherwise, the slice on Layer 5 is left blank.

  See Figure 4 for a visual help. Formally, if Σ_X and Σ_D are the alphabets of X_2 and D:

$$X_3^{\rightarrow} = \left\{ x \in (\Sigma_X \times \Sigma_D)^{\mathbb{Z}^3} : \begin{array}{l} \exists y \in X_2,\ \forall i,j,k \in \mathbb{Z},\ \pi_{1,2,3,4}(x_{i,j,k}) = y_{i,j} \\ \forall i \in \mathbb{Z},\ \exists d^i \in D,\ \forall j,k \in \mathbb{Z},\ \pi_5(x_{i,j,k}) = d^i_{j,k} \\ \forall i,j,k \in \mathbb{Z},\ \begin{array}{l} \pi_{1,2}(x_{i,j,k}) \neq (q_0, \rightarrow) \implies \pi_5(x_{i,j,k}) = \square \\ \pi_5(x_{i,j,k}) \in \{\text{╋}, \text{┃}\} \iff \pi_3(x_{i,j,k}) \in \{\ulcorner, \llcorner\} \end{array} \end{array} \right\}$$

As there exists at most a single square of infinite level in X^2_T, there exists at most a single slice $(i,\cdot,\cdot)$ of infinite level in $X_3^{\rightarrow}$.

On every slice marked by $\rightarrow$, the configuration of D breaks all periods smaller than its squares, and the size of the squares is controlled by the level of the slice in $T_{\mathcal{M}}$. Therefore:

▷ Claim 21. A configuration of $X_3^{\rightarrow}$:

1. breaks every periodicity vector $(n,\cdot,\cdot)$ and $(\cdot,n,\cdot)$ for $n \geq 1$.
2. for every slice $(i,\cdot,\cdot)$ containing $(q_0, \rightarrow)$ on the first two layers and corresponding to a column of level ℓ in X^2_T, Layer 5 breaks every vector $(\cdot,\cdot,n)$ for $1 \leq n < 2^\ell$.

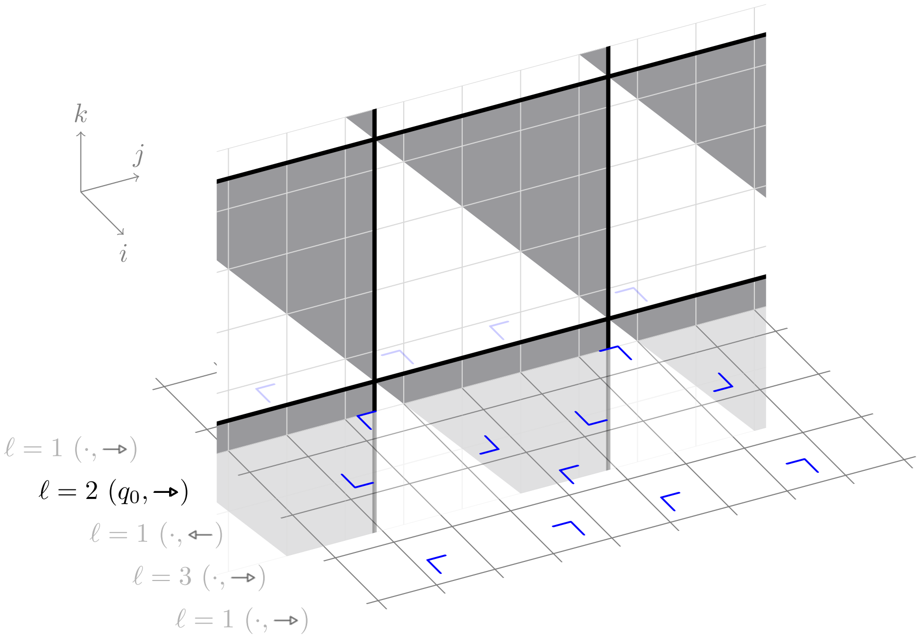

k
j
i
$\ell = 1\ (\cdot, \rightarrow)$
$\ell = 2\ (q_0, \rightarrow)$
$\ell = 1\ (\cdot, \leftarrow)$
$\ell = 3\ (\cdot, \rightarrow)$
$\ell = 1\ (\cdot, \rightarrow)$

Figure 4 A configuration of $X_3^{\rightarrow}$. The horizontal plane contains a configuration $x \in X_T^2$, and the slice of level 2 marked by $(q_0, \rightarrow)$ contains a configuration of D "synchronised" with the squares of x.

Proof.

1. X_2 is aperiodic because X_T^2 is also aperiodic, so all vectors $(n, \cdot, \cdot)$ and $(\cdot, n, \cdot)$ are broken for $n \geq 1$.
2. For a slice $(i, \cdot, \cdot)$ of level ℓ, the distance along $(0, 1, 0)$ between two consecutive lines in Layer 3 (ie. in X_T^2) is exactly 2^ℓ (see. Section 4.2, point 3 in the list of properties of configurations in X_T^2). So, Layer 5 breaks every smaller period in this direction. ◁

4.4 $\mathbb{Z}^4$ SFT X_4 and proof of the reduction

Creation of X_4

Similarly to Y_3 (Section 3.2), we build X_4 by "fusing together" $X_3^{\rightarrow}$ and $X_3^{\leftarrow}$. Formally:

$$X_4 = \{x \in (\Sigma_X \times \Sigma_D \times \Sigma_D)^{\mathbb{Z}^4} : \exists x^{\rightarrow} \in X_3^{\rightarrow}, \exists x^{\leftarrow} \in X_3^{\leftarrow},$$
$$\forall i, j, k, l \in \mathbb{Z},\ \pi_{1,2,3,4,5}(x_{i,j,k,l}) = x^{\rightarrow}_{i,j,l} \text{ and } \pi_{1,2,3,4,6}(x_{i,j,k,l}) = x^{\leftarrow}_{i,j,k}\}$$

▷ Claim 22. X_4 is an SFT.

Proof. Both $X_3^{\rightarrow}$ and $X_3^{\leftarrow}$ are SFTs, since X_2 is an SFT. ◁

Reduction RS $\leq$ AD

▶ **Lemma 23.** *There exists an aperiodic configuration in X_4 if and only if there exists a run of $\mathcal{M}$ in which q_0 occurs infinitely often.*

Proof. This is the same proof as for Lemma 16, except that vectors along the first two dimensions are broken by the Toeplitz structure on layer 3. Otherwise, Layers 5 and 6 break every vector $(0, 0, \cdot, \cdot)$ if and only if the run visits q_0 infinitely often. ◀

This concludes the proof of Theorem 17.

5 Complexity and aperiodic configurations

The *complexity function* of a $\mathbb{Z}^d$ subshift X is $N_X(n) = \#\{w \in \Sigma^{[\![0,n-1]\!]^d} : \exists x \in X, w \sqsubseteq x\}$. In this section, we see that subshifts of high complexity are expected to have aperiodic configurations.

▶ **Definition 24** (Dimensional entropy). *Define $h_k(X)$, the entropy of dimension k, as:*

$$h_k(X) = \limsup_{n\to+\infty} \frac{\log N_X(n)}{n^k} \in [0, +\infty]$$

[8, Theorem 10] proves that a $\mathbb{Z}^2$ subshift X with no aperiodic configurations is almost topologically conjugated to (i.e. “nearly behaves as”) a $\mathbb{Z}$ subshift of the same type. In particular, $h_1(X) = +\infty$ is only possible when X contains an aperiodic configuration.

[11, Corollary 13] entails that $h_d(X) > 0$ for a $\mathbb{Z}^d$ SFT implies the existence of aperiodic configurations in any dimension. We improve this result as follows:

▶ **Proposition 25.** *Let X be a $\mathbb{Z}^d$ SFT or sofic subshift. If $h_{d-1}(X) = +\infty$, then there exists an aperiodic configuration in X.*

Proof. Let $X' \subseteq \Sigma'^{\mathbb{Z}^d}$ be a SFT cover of X: $\pi(X') = X$ for π a letter-to-letter projection. W.l.o.g., assume X' is defined by adjacency constraints. Consider all patterns on $[\![0, n-1]\!]^d$ admissible in X' : they have exactly $N_X(n)$ different projections by π, but the number $N^b_{X'}(n)$ of different patterns on the boundary of $[\![0, n-1]\!]^d$ is at most $\Sigma'^{2dn^{d-1}}$. As $\log N_X(n)/n^{d-1}$ is unbounded, there exists some n such that:

$$2d(\log \Sigma') < \frac{\log N_X(n)}{n^{d-1}} \iff \Sigma'^{2dn^{d-1}} < N_X(n) \implies N^b_{X'}(n) < N_X(n)$$

By the pigeonhole principle, there exists a pattern b on the boundary admissible in X' that can be extended on the cube in two different admissible patterns b^+, b^- such that $\pi(b^+) \neq \pi(b^-)$. Consider a configuration $x' \in X'$ in which b^+ appears. If $\pi(x')$ is not already aperiodic, swapping b^+ for b^- in x' at a some arbitrary position leads to a configuration $\pi(x'') \in X$ which is aperiodic. ◀

Proposition 25 is tight: the $\mathbb{Z}^d$-lift of a $\mathbb{Z}^{d-1}$ SFT ($d \geq 3$) is periodic by definition, and can have arbitrarily high entropy of dimension $(d-1)$. We conjecture that Proposition 25 holds for all subshifts even in dimension $d > 2$.

Proposition 25 shows that **AD** is a problem that is only relevant for low complexity subshifts, which is where its full computational complexity “lies”. Indeed, the problem of deciding whether $h_{d-1}(X) = +\infty$ is Π^0_3 (it is equivalent to $\forall k\ \exists n\ N_X(n) > k$, and $N_X(n)$ is a Π^0_1 integer), which is much easier than the Σ^1_1-completeness of **AD** on $\mathbb{Z}^3$ sofic subshifts.

6 Open problems

The main remaining question is, of course, the case of $\mathbb{Z}^3$ SFTs. The method we developed above to prove Σ^1_1-completeness in the case of $\mathbb{Z}^3$ sofic subshifts cannot be applied. Indeed, embedding computations in an SFT requires at least two aperiodic dimensions; and we need two other dimensions (because of the positions of level ∞) which can be periodic or aperiodic.

We conjecture that aperiodic configurations in $\mathbb{Z}^3$ SFTs behave similarly as in $\mathbb{Z}^2$ subshifts: each $\mathbb{Z}^3$ SFT containing aperiodic configurations seems to have “centers” of aperiodicity, i.e. concentric zones in which periods are broken. The distance from the center might depend on the length of the vector, $|\Sigma|$ and the size of the largest forbidden pattern.

However, there seem to be important differences. First, for $\mathbb{Z}^3$-SFT, not all aperiodic configurations have a center of aperiodicity in their orbit closure: this center may be in an unrelated aperiodic configuration. Second, results of [8] are valid for all $\mathbb{Z}^2$ subshifts, but our conjecture must be specific to $\mathbb{Z}^3$ SFTs, and a proof requires SFT-specific techniques.

Considering subshifts on more general groups (other than $\mathbb{Z}^d$), there is an active research theme looking for conditions on groups which make the Domino problem undecidable. In this context, we would like to obtain conditions that make **AD** Π^0_1- or Σ^1_1-complete.

References

1 Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *Acta Applicandae Mathematicae*, 126:35–63, 2013. `doi:10.1007/s10440-013-9808-5`.

2 Robert Berger. *The Undecidability of the Domino Problem.* Number 66 in Memoirs of the American Mathematical Society. American Mathematical Society, 1966.

3 Valérie Berthé and Michel Rigo. *Combinatorics, Words and Symbolic Dynamics.* Number 159 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2016. `doi:10.1017/CBO9781139924733`.

4 Antonin Callard and Pascal Vanier. Computational Characterization of Surface Entropies for $\mathbb{Z}^2$ Subshifts of Finite Type. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 122:1–122:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.ICALP.2021.122`.

5 Bruno Durand, Andrei E. Romashchenko, and Alexander Shen. Effective closed subshifts in 1D can be implemented in 2D. In Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig, editors, *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *Lecture Notes in Computer Science*, pages 208–226. Springer, 2010. `doi:10.1007/978-3-642-15025-8_12`.

6 Shmuel Friedland. On the entropy of $\mathbb{Z}^d$ subshifts of finite type. *Linear Algebra and its Applications*, 252(1):199–220, 1997. `doi:10.1016/0024-3795(95)00676-1`.

7 Silvère Gangloff and Benjamin Hellouin de Menibus. Effect of quantified irreducibility on the computability of subshift entropy. *Discrete & Continuous Dynamical Systems - A*, 39(4):1975–2000, 2019. `doi:10.3934/dcds.2019083`.

8 Anaël Grandjean, Benjamin Hellouin de Menibus, and Pascal Vanier. Aperiodic points in $\mathbb{Z}^2$-subshifts. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 128:1–128:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.ICALP.2018.128`.

9 David Harel. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *Journal of the ACM*, 33(1):224–248, 1986. `doi:10.1145/4904.4993`.

10 Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones mathematicae*, 176:131–167, 2008. `doi:10.1007/s00222-008-0161-7`.

11 Michael Hochman. On the automorphism groups of multidimensional shifts of finite type. *Ergodic Theory and Dynamical Systems*, 30:809–840, 2009. `doi:10.1017/S0143385709000248`.

12 Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics*, 171(3):2011–2038, 2010. `doi:10.4007/annals.2010.171.2011`.

13 Konrad Jacobs and Michael Keane. 0–1–sequences of Toeplitz type. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 13(2):123–131, 1969. `doi:10.1007/BF00537017`.

14 Emmanuel Jeandel and Pascal Vanier. *The Undecidability of the Domino Problem*, pages 293–357. Springer, 2020. `doi:10.1007/978-3-030-57666-0_6`.

15 Tom Meyerovitch. Growth-type invariants for $\mathbb{Z}^d$ subshifts of finite type and arithmetical classes of real numbers. *Inventiones mathematicae*, 184:567–589, 2011. `doi:10.1007/s00222-010-0296-1`.

16 Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *Journal d'Analyse Mathématique*, 53(1):139–186, 1989. `doi:10.1007/BF02793412`.

17 Piergiorgio Odifreddi. *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers.* Elsevier, 1989.

18 Ronnie Pavlov and Michael Schraudner. Entropies realizable by block gluing $\mathbb{Z}^d$ shifts of finite type. *Journal d'Analyse Mathématique*, 126(1):113–174, 2015. `doi:10.1007/s11854-015-0014-4`.

19 Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones mathematicae*, 12:177–209, 1971. `doi:10.1007/BF01418780`.

20 Robert I. Soare. *Turing Computability: Theory and Applications.* Theory and Applications of Computability. Springer, 1st edition, 2016. `doi:10.1007/978-3-642-31933-4`.

21 Linda Brown Westrick. Seas of squares with sizes from a Π^0_1 set. *Israel Journal of Mathematics*, 222:431–462, 2017. `doi:10.1007/s11856-017-1596-6`.

Symmetry and Quantum Query-To-Communication Simulation

Sourav Chakraborty ✉
Indian Statistical Institute, Kolkata, India

Arkadev Chattopadhyay ✉
TIFR, Mumbai, India

Peter Høyer ✉
Department of Computer Science, University of Calgary, Canada

Nikhil S. Mande[1] ✉
CWI, Amsterdam, The Netherlands

Manaswi Paraashar ✉
Indian Statistical Institute, Kolkata, India

Ronald de Wolf ✉
QuSoft, CWI, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Abstract

Buhrman, Cleve and Wigderson (STOC'98) showed that for every Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ and $G \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$, the bounded-error quantum communication complexity of the composed function $f \circ G$ equals $O(\mathsf{Q}(f) \log n)$, where $\mathsf{Q}(f)$ denotes the bounded-error quantum query complexity of f. This is achieved by Alice running the optimal quantum query algorithm for f, using a round of $O(\log n)$ qubits of communication to implement each query. This is in contrast with the classical setting, where it is easy to show that $\mathsf{R}^{cc}(f \circ G) \le 2\mathsf{R}(f)$, where R^{cc} and R denote bounded-error communication and query complexity, respectively. Chakraborty et al. (CCC'20) exhibited a total function for which the $\log n$ overhead in the BCW simulation is required. This established the somewhat surprising fact that quantum reductions are in some cases inherently more expensive than classical reductions. We improve upon their result in several ways.

- We show that the $\log n$ overhead is *not* required when f is symmetric (i.e., depends only on the Hamming weight of its input), generalizing a result of Aaronson and Ambainis for the Set-Disjointness function (Theory of Computing'05). Our upper bound assumes a shared entangled state, though for most symmetric functions the assumed number of entangled qubits is less than the communication and hence could be part of the communication.
- In order to prove the above, we design an efficient distributed version of noisy amplitude amplification that allows us to prove the result when f is the OR function. This also provides a different, and arguably simpler, proof of Aaronson and Ambainis's $O(\sqrt{n})$ communication upper bound for Set-Disjointness.
- In view of our first result above, one may ask whether the $\log n$ overhead in the BCW simulation can be avoided even when f is transitive, which is a weaker notion of symmetry. We give a strong negative answer by showing that the $\log n$ overhead is still necessary for some transitive functions even when we allow the quantum communication protocol an error probability that can be arbitrarily close to 1/2 (this corresponds to the unbounded-error model of communication).
- We also give, among other things, a general recipe to construct functions for which the $\log n$ overhead is required in the BCW simulation in the bounded-error communication model, even if the parties are allowed to share an arbitrary prior entangled state for free.

[1] Part of this work was done while the author was a postdoc at Georgetown University.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 20; pp. 20:1–20:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Quantum complexity theory; Theory of computation → Communication complexity; Theory of computation → Oracles and decision trees

Keywords and phrases Classical and quantum communication complexity, query-to-communication-simulation, quantum computing

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.20

Related Version *Full Version*: https://arxiv.org/abs/2012.05233

Funding *Arkadev Chattopadhyay*: Partially supported by a MATRICS grant of the Science and Enginnering Research Board, DST, India.
Nikhil S. Mande: Supported by the Dutch Research Council (NWO) through QuantERA project QuantAlgo 680-91-034.
Ronald de Wolf: Partially supported by ERC Consolidator Grant 615307-QPROGRESS (which ended February 2019), and by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037), and through QuantERA ERA-NET Cofund project QuantAlgo (680-91-034).

1 Introduction

1.1 Motivation and main results

The classical model of communication complexity was introduced by Yao [24], who also subsequently introduced its quantum analogue [25]. Communication complexity has important applications in several disciplines, in particular for lower bounds on circuits, data structures, streaming algorithms, and many other complexity measures (see, for example, [16] and the references therein).

A natural way to derive a communication problem from a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ is via composition. Let $f : \{-1,1\}^n \to \{-1,1\}$ be a function and let $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$ be a "two-party function". Then $F = f \circ G : \{-1,1\}^{nj} \times \{-1,1\}^{nk} \to \{-1,1\}$ denotes the function corresponding to the communication problem in which Alice is given input $X = (X_1,\dots,X_n) \in \{-1,1\}^{nj}$, Bob is given $Y = (Y_1,\dots,Y_n) \in \{-1,1\}^{nk}$, and their task is to compute $F(X,Y) = f(G(X_1,Y_1),\dots,G(X_n,Y_n))$. Many well-known functions in communication complexity are derived in this way, such as Set-Disjointness ($\mathsf{DISJ}_n := \mathsf{NOR}_n \circ \mathsf{AND}_2$), Inner Product ($\mathsf{IP}_n := \mathsf{PARITY}_n \circ \mathsf{AND}_2$) and Equality ($\mathsf{EQ}_n := \mathsf{NOR}_n \circ \mathsf{XOR}_2$). A natural approach to obtain efficient quantum communication protocols for $f \circ G$ is to "simulate" a quantum query algorithm for f, where a query to the ith input bit of f is simulated by a communication protocol that computes $G(X_i,Y_i)$. Buhrman, Cleve and Wigderson [7] observed that such a simulation is indeed possible if G is bitwise AND or XOR.

▶ **Theorem 1** ([7]). *For every Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ and $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$, we have*

$$\mathsf{Q}^{cc}(f \circ \square) = O(\mathsf{Q}(f)\log n).$$

Here $\mathsf{Q}(f)$ denotes the bounded-error quantum query complexity of f, and $\mathsf{Q}^{cc}(f \circ \square)$ denotes the bounded-error quantum communication complexity for computing $f \circ \square$. Throughout this paper, we refer to Theorem 1 as the BCW simulation. [7] used this, for instance, to show that the bounded-error quantum communication complexity of the Set-Disjointness function is $O(\sqrt{n}\log n)$, using Grover's $O(\sqrt{n})$-query search algorithm [11] for the NOR_n function.

It is folklore in the classical world that the analogous simulation does not incur a $\log n$ factor overhead. That is,

$$\mathsf{R}^{cc}(f \circ \square) \leq 2\mathsf{R}(f), \tag{1}$$

where $\mathsf{R}(f)$ denotes the bounded-error randomized query complexity of f and $\mathsf{R}^{cc}(f \circ \square)$ denotes the bounded-error randomized communication complexity for computing $f \circ \square$. Thus, a natural question is whether the multiplicative $\log n$ blow-up in the communication cost in the BCW simulation is necessary. Chakraborty et al. [9] answered this question and exhibited a total function for which the $\log n$ blow-up is indeed necessary when XOR_2 is the inner function.

▶ **Theorem 2** ([9, Theorem 2]). *There exists a function $f : \{-1,1\}^n \to \{-1,1\}$ such that*

$$\mathsf{Q}^{cc,*}(f \circ \mathsf{XOR}_2) = \Omega(\mathsf{Q}(f) \log n).$$

Here $\mathsf{Q}^{cc,*}(F)$ denotes the bounded-error quantum communication complexity of two-party function F when Alice and Bob shared an entangled state at the start of the protocol for free. Comparing Theorem 2 with Equation 1 we see the somewhat surprising fact that quantum reductions can in some cases be more expensive than classical reductions.

This gives rise to the following basic question: is there a natural class of functions for which the $\log n$ overhead in the BCW simulation is *not* required? Improving upon Høyer and de Wolf [13], Aaronson and Ambainis [1] showed that for the canonical problem of Set-Disjointness, the $\log n$ overhead in the BCW simulation can be avoided. Since the outer function NOR_n is symmetric (i.e., it only depends on the Hamming weight of its input, its number of -1s), a natural question is whether the $\log n$ overhead can be avoided whenever the outer function is symmetric. Our first result gives a positive answer to this question.

▶ **Theorem 3.** *For every symmetric Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ and two-party function $G : \{-1,1\}^j \times \{-1,1\}^k \to \{0,1\}$, we have*

$$\mathsf{Q}^{cc,*}(f \circ G) = O(\mathsf{Q}(f)\mathsf{Q}_E^{cc}(G)).$$

Here $\mathsf{Q}_E^{cc}(G)$ denotes the *exact* quantum communication complexity of G, where the error probability is 0. In particular, if $G \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$ then $\mathsf{Q}_E^{cc}(G) = 1$ and hence $\mathsf{Q}^{cc,*}(f \circ G) = O(\mathsf{Q}(f))$.

▶ Remark 4. If $\mathsf{Q}(f) = \Theta(\sqrt{tn})$, then our protocol in the proof of Theorem 3 starts from a shared entangled state of $O(t \log n)$ EPR-pairs. Note that if $t \leq n\mathsf{Q}_E^{cc}(G)^2/(\log n)^2$ (this condition holds for instance if $\mathsf{Q}_E^{cc}(G) \geq \log n$) then this number of EPR-pairs is no more than the amount of communication and hence might as well be established in the first message, giving asymptotically the same upper bound $\mathsf{Q}^{cc}(f \circ G) = O(\mathsf{Q}(f)\mathsf{Q}_E^{cc}(G))$ for the model without prior entanglement.

The next question one might ask is whether one can weaken the notion of symmetry required in Theorem 3. A natural generalization of the class of symmetric functions is the class of *transitive-symmetric* functions. A function $f : \{-1,1\}^n \to \{-1,1\}$ is said to be transitive-symmetric if for all $i, j \in [n]$, there exists $\sigma \in S_n$ such that $\sigma(i) = j$, and $f(x) = f(\sigma(x))$ for all $x \in \{-1,1\}^n$. Here, and in the rest of the paper, by $\sigma(x)$ we mean the n-bit string $x_{\sigma(1)}, \ldots, x_{\sigma(n)}$. Henceforth we refer to transitive-symmetric functions as simply transitive functions. Can the $\log n$ overhead in the BCW simulation be avoided whenever the outer function is transitive? We give a negative answer to this question in a strong sense: the $\log n$ overhead is still necessary even when we allow the quantum communication protocol an error probability that can be arbitrarily close to $1/2$.

▶ **Theorem 5.** *There exists a transitive and total function $f : \{-1,1\}^n \to \{-1,1\}$, such that*

$$\mathsf{UPP}^{cc}(f \circ \square) = \Omega(\mathsf{Q}(f) \log n)$$

for every $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$.

Here $\mathsf{UPP}^{cc}(f \circ \square)$ denotes the unbounded-error quantum communication complexity of $f \circ \square$ (adding "quantum" here only changes the communication complexity by a constant factor). The unbounded-error model of communication was introduced by Paturi and Simon [21] and is the strongest communication complexity model against which we know how to prove explicit lower bounds. This model is known to be strictly stronger than the bounded-error quantum model. For instance, the Set-Disjointness function on n inputs requires $\Omega(n)$ bits or $\Omega(\sqrt{n})$ qubits of communication in the bounded-error model, but only requires $O(\log n)$ bits of communication in the unbounded-error model. In fact, it follows from a recent result of Hatami, Hosseini and Lovett [12] that there exists a function $F : \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ with $\mathsf{Q}^{cc,*}(F) = \Omega(n)$ while $\mathsf{UPP}^{cc}(F) = O(1)$.

Theorem 3 and Theorem 5 clearly demonstrate the role of symmetry in determining the presence of the $\log n$ overhead in the BCW query-to-communication simulation: this overhead is absent for symmetric functions (Theorem 3), but present for a transitive function even when the model of communication under consideration is as strong as the unbounded-error model (Theorem 5). We also give a general recipe to construct functions for which the $\log n$ overhead is required in the BCW simulation in the bounded-error communication model (see Theorem 6).

1.2 Overview of our approach and techniques

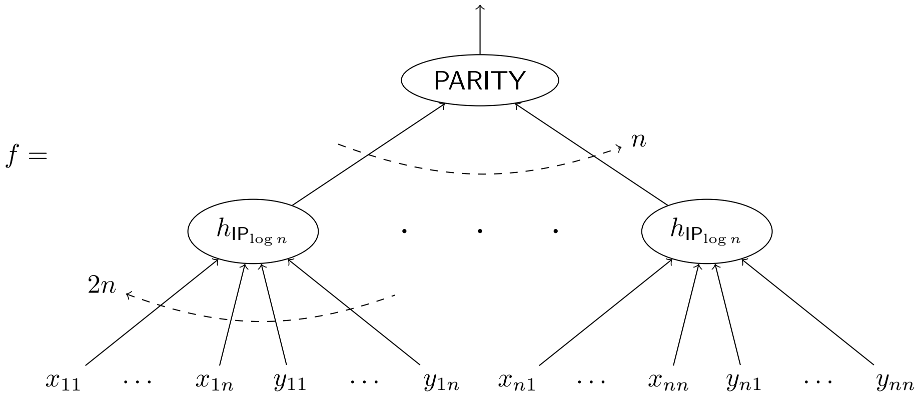
In this section we discuss the ideas that go into the proofs of Theorem 3 and Theorem 5.

1.2.1 Communication complexity upper bound for symmetric functions

To prove Theorem 3 we use the well-known fact that every symmetric function f has an interval around Hamming weight $n/2$ where the function is constant; for NOR_n the length of this interval would be essentially n, while for PARITY_n it would be 1. To compute f, it suffices to either determine that the Hamming weight of the input lies in that interval (because the function value is the same throughout that interval) or to count the Hamming weight exactly.

For two-party functions of the form $f \circ G$, we want to do this type of counting on the n-bit string $z = (G(X_1, Y_1), \ldots, G(X_n, Y_n)) \in \{-1,1\}^n$. We show how this can be done with $O(\mathsf{Q}(f)\, \mathsf{Q}_E^{cc}(G))$ qubits of communication if we had a quantum protocol that can find -1s in the string z at a cost of $O(\sqrt{n}\, \mathsf{Q}_E^{cc}(G))$ qubits. Such a protocol was already given by Aaronson and Ambainis for the special case where $G = \mathsf{AND}_2$ for their optimal quantum protocol for Set-Disjointness, as a corollary of their quantum walk algorithm for search on a grid [1]. In this paper we give an alternative $O(\sqrt{n}\, \mathsf{Q}_E^{cc}(G))$-qubit protocol. This implies the result of Aaronson and Ambainis as a special case, but it is arguably simpler and may be of independent interest.

Our protocol can be viewed as an efficient distributed implementation of amplitude amplification with faulty components. In particular, we replace the usual reflection about the uniform superposition by an imperfect reflection about the n-dimensional maximally entangled state (= $\log n$ EPR-pairs if n is a power of 2). Such a reflection would require $O(\log n)$ qubits of communication to implement perfectly, but can be implemented with small



$f =$
PARITY
n
$h_{\mathsf{IP}_{\log n}}$
$h_{\mathsf{IP}_{\log n}}$
$2n$
x_{11} $\cdots$ x_{1n} y_{11} $\cdots$ y_{1n} x_{n1} $\cdots$ x_{nn} y_{n1} $\cdots$ y_{nn}

Figure 1 If for all $j \in [n]$ and some $s_j, t_j \in \{-1,1\}^{\log n}$, the inputs to the j-th $h_{\mathsf{IP}_{\log n}}$ are Hadamard codewords in $\pm H(s_j)$ and $\pm H(t_j)$, then $f = \mathsf{PARITY}(\mathsf{IP}_{\log n}(s_1,t_1),\ldots,\mathsf{IP}_{\log n}(s_n,t_n))$. If there exists at least one $j \in [n]$ for which either $x_{j1},\ldots,x_{jn}$ or $y_{j1},\ldots,y_{jn}$ is not a Hadamard codeword, then f outputs -1. This function f equals $\mathsf{PARITY}_n \,\widetilde{\circ}\, h_{\mathsf{IP}_{\log n}}$ (see Definition 26 and Definition 28).

error using only $O(1)$ qubits of communication, by invoking the efficient protocol of Aharonov et al. [2, Theorem 1] that tests whether a given bipartite state equals the n-dimensional maximally entangled state. Still, at the start of this protocol we need to assume (or establish by means of quantum communication) a shared state of $\log n$ EPR-pairs. If $\mathsf{Q}(f) = \Theta(\sqrt{tn})$ then our protocol for $f \circ G$ will run the -1-finding protocol $O(t)$ times, which accounts for our assumption that we share $O(t \log n)$ EPR-pairs at the start of the protocol.

1.2.2 Communication complexity lower bound for transitive functions

For proving Theorem 5, we exhibit a transitive function $f : \{-1,1\}^{2n^2} \to \{-1,1\}$ whose bounded-error quantum query complexity is $O(n)$ and the unbounded-error communication complexity of $f \circ \square$ is $\Omega(n \log n)$ for $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$.

Function construction and transitivity. For the construction of f we first require the definition of Hadamard codewords. The Hadamard codeword of $s \in \{-1,1\}^{\log n}$, denoted by $H(s) \in \{-1,1\}^n$, is a list of all parities of s. That is, $(H(s))_t = \prod_{i:s_i=-1} t_i$ for all $t \in \{-1,1\}^{\log n}$. See Figure 1 for a graphical visualization of f.

Using properties of IP and Hadamard codewords, and the symmetry of PARITY_n, we are able to show that f is transitive (see Claim 32).

Query upper bound. The query upper bound of $O(n)$ follows along the lines of [9], using the Bernstein-Vazirani algorithm to decode the Hadamard codewords, and Grover's algorithm to check that they actually are Hadamard codewords. This approach was in turn inspired by a query upper bound due to Ambainis and de Wolf [3]. See the proof of Theorem 29 for the query algorithm and its analysis.

Communication lower bound. Towards the unbounded-error communication lower bound, we first recall that each input block of f equals $\mathsf{IP}_{\log n}$ if the inputs to each block are promised to be Hadamard codewords. Hence f equals $\mathsf{IP}_{n \log n}$ under this promise, since $\mathsf{PARITY}_n \circ \mathsf{IP}_{\log n} = \mathsf{IP}_{n \log n}$. Thus by setting certain inputs to Alice and Bob suitably, $f \circ \square$

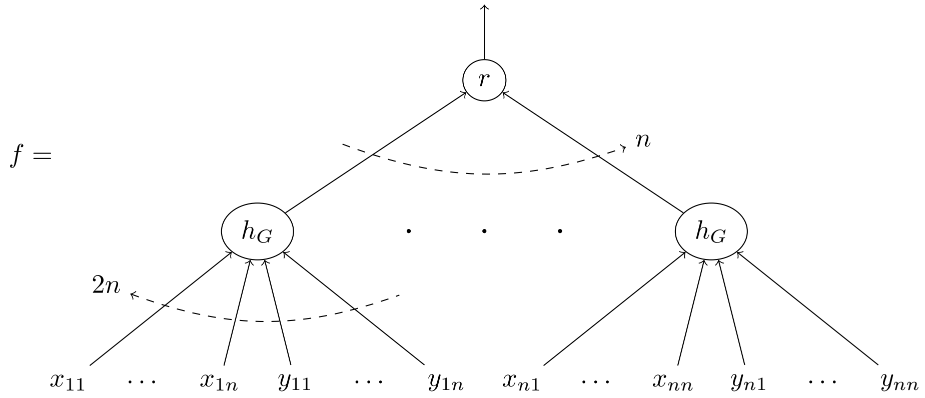


Figure 2 In this figure, $G : \{-1,1\}^{\log} \times \{-1,1\}^{\log n} \to \{-1,1\}$. If for all $j \in [n]$ and some $s_j, t_j \in \{-1,1\}^{\log n}$, the inputs to the j-th h_G are Hadamard codewords in $\pm H(s_j)$ and $\pm H(t_j)$, then $f = r(G(s_1,t_1),\ldots,G(s_n,t_n))$. If there exists at least one $j \in [n]$ for which either $x_{j1},\ldots,x_{jn}$ or $y_{j1},\ldots,y_{jn}$ is not a Hadamard codeword, then f outputs -1. This function f equals $r \mathbin{\widetilde{\circ}} h_G$ (see Definition 26 and Definition 28).

is at least as hard as $\mathsf{IP}_{n\log n}$ for $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$ (for a formal statement, see Lemma 31 with $r = \mathsf{PARITY}_n$ and $g = \mathsf{IP}_{\log n}$). It is known from a seminal result of Forster [10] that the unbounded-error communication complexity of $\mathsf{IP}_{n\log n}$ equals $\Omega(n\log n)$, completing the proof of the lower bound. This proof is more general than and arguably simpler than the proof of the lower bound for bounded-error quantum communication complexity in [9, Theorem 2].

1.3 Other results

We give a general recipe for constructing a class of functions that witness tightness of the BCW simulation where the inner gadget is either AND_2 or XOR_2. However, the communication lower bound we obtain here is in the bounded-error model in contrast to Theorem 5, where the communication lower bound is proven in the unbounded-error model.

The functions f constructed for this purpose are composed functions similar to the construction in Figure 1, except that we are able to use a more general class of functions in place of the outer PARITY function, and also a more general class of functions in place of the inner $\mathsf{IP}_{\log n}$ functions. See Figure 2 and its caption for an illustration and a more precise definition.

We require some additional constraints on the outer and inner functions. First, the approximate degree of r should be $\Omega(n)$. Second, the discrepancy of G should be small with respect to some "balanced" probability distribution (see Definition 17 and Definition 16 for formal definitions of these notions).

▶ **Theorem 6.** *Let $r : \{-1,1\}^n \to \{-1,1\}$ be such that $\widetilde{\deg}(r) = \Omega(n)$ and let $G : \{-1,1\}^{\log n} \times \{-1,1\}^{\log n} \to \{-1,1\}$ be a total function. Define $f : \{-1,1\}^{2n^2} \to \{-1,1\}$ as in Figure 2. If there exists $\mu : \{-1,1\}^{\log n} \times \{-1,1\}^{\log n} \to \mathbb{R}$ that is a balanced probability distribution with respect to G and $\mathrm{disc}_\mu(G) = n^{-\Omega(1)}$, then for every $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$,*

$$\mathsf{Q}(f) = O(n), \qquad \text{and} \qquad \mathsf{Q}^{cc,*}(f \circ \square) = \Omega(n\log n).$$

The query upper bound follows along similar lines as that of Theorem 5. For the lower bound, we first show via a reduction that for f as described in Figure 2 and $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$, the communication problem $f \circ \square$ is at least as hard as $r \circ G$ (see Lemma 31).

This part of the lower bound proof is the same as in the proof of Theorem 5. For the hardness of $r \circ G$ (which in the case of Theorem 5 turned out to be $\mathsf{IP}_{n \log n}$, for which Forster's theorem yields an unbounded-error communication lower bound), we are able to use a theorem implicit in a work of Lee and Zhang [17]. This theorem gives a lower bound on the bounded-error communication complexity of $r \circ G$ in terms of the approximate degree of r and the discrepancy of G under a balanced distribution. Due to space constraints we defer the proof of Theorem 6 to the full version of our paper [8].

We recover the result of Chakraborty et al. (Theorem 2) using a more general technique, and additionally show that $\mathsf{Q}^{cc,*}(f \circ \mathsf{AND}_2) = \Omega(\mathsf{Q}(f) \log n)$, where f is as in Theorem 2. We refer the reader to the full version [8] for details.

1.4 Organization

Section 2 gives notations and preliminaries. In Section 3 we prove Theorem 3, which shows that the $\log n$ overhead in the BCW simulation can be avoided when the outer function is symmetric. This proof relies on our new one-sided error protocol for finding solutions in the string $z = (G(X_1, Y_1), \ldots, G(X_n, Y_n)) \in \{-1, 1\}^n$, as a corollary of our distributed version of amplitude amplification. We give this protocol in Appendix A.

We prove Theorem 5 in Section 4. This is our result regarding necessity of the $\log n$ overhead in the BCW simulation in the unbounded-error model of communication.

2 Notation and preliminaries

Without loss of generality, we assume n to be a power of 2 in this paper, unless explicitly stated otherwise. All logarithms in this paper are base 2. Let S_n denote the symmetric group over the set $[n] = \{1, \ldots, n\}$. For a string $x \in \{-1, 1\}^n$ and $\sigma \in S_n$, let $\sigma(x)$ denote the string $x_{\sigma(1)}, \ldots, x_{\sigma(n)} \in \{-1, 1\}^n$. Consider an arbitrary but fixed bijection between subsets of $[\log n]$ and elements of $[n]$. For a string $s \in \{-1, 1\}^{\log n}$, we abuse notation and also use s to denote the equivalent element of $[n]$. The view we take will be clear from context. For a string $x \in \{-1, 1\}^n$ and set $S \subseteq [n]$, define the string $x_S \in \{-1, 1\}^S$ to be the restriction of x to the coordinates in S. Let 1^n and $(-1)^n$ denote the n-bit string $(1, 1, \ldots, 1)$ and $(-1, -1, \ldots, -1)$, respectively.

2.1 Boolean functions

For two bits $b_1, b_2 \in \{-1, 1\}$, let $b_1 \wedge b_2$ be defined to be -1 if $b_1 = b_2 = -1$, and 1 otherwise. For strings $x, y \in \{-1, 1\}^n$, let $\langle x, y \rangle$ denote the inner product (mod 2) of x and y. That is, $\langle x, y \rangle = \prod_{i=1}^{n} (x_i \wedge y_i)$. For every positive integer n, let $\mathsf{PARITY}_n : \{-1, 1\}^n \to \{-1, 1\}$ be defined as:

$$\mathsf{PARITY}_n(x_1, \ldots, x_n) = \prod_{i \in [n]} x_i.$$

▶ **Definition 7** (Symmetric functions). *A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is symmetric if for all $\sigma \in S_n$ and for all $x \in \{-1, 1\}^n$ we have $f(x) = f(\sigma(x))$.*

▶ **Definition 8** (Transitive functions). *A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is transitive if for all $i, j \in [n]$ there exists a permutation $\sigma \in S_n$ such that*

- *$\sigma(i) = j$, and*
- *$f(x) = f(\sigma(x))$ for all $x \in \{-1, 1\}^n$.*

▶ **Definition 9** (Approximate degree). *For every $\varepsilon \geq 0$, the ε-approximate degree of a function $f : \{-1,1\}^n \to \{-1,1\}$ is defined to be the minimum degree of a real polynomial $p : \{-1,1\}^n \to \mathbb{R}$ that uniformly approximates f to error ε. That is,*

$$\widetilde{\deg}_\varepsilon(f) = \min\left\{\deg(p) : |p(x) - f(x)| \leq \varepsilon \text{ for all } x \in \{-1,1\}^n\right\}.$$

Unless specified otherwise, we drop ε from the subscript and assume $\varepsilon = 1/3$.

We assume familiarity with quantum computing [19], and use $\mathsf{Q}_\varepsilon(f)$ to denote the ε-error query complexity of f. Unless specified otherwise, we drop ε from the subscript and assume $\varepsilon = 1/3$.

▶ **Theorem 10** ([4]). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a function. Then $\mathsf{Q}(f) \geq \widetilde{\deg}(f)/2$.*

2.2 Communication complexity

We assume familiarity with communication complexity [16].

▶ **Definition 11** (Two-party function). *We call a function $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$ a* two-party function *to indicate that it corresponds to a communication problem in which Alice is given input $x \in \{-1,1\}^j$, Bob is given input $y \in \{-1,1\}^k$, and their task is to compute $G(x,y)$.*

▶ Remark 12. Throughout this paper, we use uppercase letters to denote two-party functions, and lowercase letters to denote functions which are not two-party functions.

▶ **Definition 13** (Composition with two-party functions). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a function and let $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$ be a two-party function. Then $F = f \circ G : \{-1,1\}^{nj} \times \{-1,1\}^{nk} \to \{-1,1\}$ denotes the two-party function corresponding to the communication problem in which Alice is given input $X = (X_1, \ldots, X_n) \in \{-1,1\}^{nj}$, Bob is given $Y = (Y_1, \ldots, Y_n) \in \{-1,1\}^{nk}$, and their task is compute $F(X,Y) = f(G(X_1,Y_1), \ldots, G(X_n,Y_n))$.*

▶ **Definition 14** (Inner Product function). *For every positive integer n, define the function $\mathsf{IP}_n : \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ by $\mathsf{IP}_n(x,y) = \langle x, y\rangle$. In other words, $\mathsf{IP}_n = \mathsf{PARITY}_n \circ \mathsf{AND}_2$.*

▶ **Observation 15.** *For all positive integers k, t, $\mathsf{PARITY}_k \circ \mathsf{IP}_t = \mathsf{IP}_{kt}$.*

We also assume familiarity with quantum communication complexity [23]. We use $\mathsf{Q}^{cc}_\varepsilon(G)$ and $\mathsf{Q}^{cc,*}_\varepsilon(G)$ to represent the ε-error quantum communication complexity of a two-party function G in the models without and with unlimited shared entanglement, respectively. Unless specified otherwise, we drop ε from the subscript and assume $\varepsilon = 1/3$.

▶ **Definition 16** (Balanced probability distribution). *We call a probability distribution $\mu : \{-1,1\}^n \to \mathbb{R}$ balanced w.r.t. a function $f : \{-1,1\}^n \to \{-1,1\}$ if $\sum_{x \in \{-1,1\}^n} f(x)\mu(x) = 0$.*

▶ **Definition 17** (Discrepancy). *Let $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$ be a function and λ be a distribution on $\{-1,1\}^j \times \{-1,1\}^k$. For every $S \subseteq \{-1,1\}^j$ and $T \subseteq \{-1,1\}^k$, define*

$$\mathrm{disc}_\lambda(S \times T, G) = \left|\sum_{x,y \in S \times T} G(x,y)\lambda(x,y)\right|.$$

The discrepancy of G under the distribution λ is defined to be

$$\mathrm{disc}_\lambda(G) = \max_{S \subseteq \{-1,1\}^j, T \subseteq \{-1,1\}^k} \mathrm{disc}_\lambda(S \times T, G),$$

and the discrepancy of f is defined to be $\mathrm{disc}(G) = \min_\lambda \mathrm{disc}_\lambda(G)$.

2.3 Additional concepts from quantum computing

The Bernstein-Vazirani algorithm [5] is a quantum query algorithm that takes an n-bit string as input and outputs a $(\log n)$-bit string. The algorithm has the following properties:

- the algorithm makes one quantum query to the input and
- if the input $x \in \{-1,1\}^n$ satisfies $x \in \pm H(s)$ for some $s \in \{-1,1\}^{\log n}$, then the algorithm returns s with probability 1.

Consider a symmetric Boolean function $f : \{-1,1\}^n \to \{-1,1\}$. Define the quantity

$$\Gamma(f) = \min\{|2k-n+1| : f(x) \neq f(y) \text{ if } |x| = k \text{ and } |y| = k+1\}$$

from [20]. One can think of $\Gamma(f)$ as essentially the length of the interval of Hamming weights around $n/2$ where f is constant (for example, for the majority and parity functions this would be 1, and for OR_n this would be $n-1$).

▶ **Theorem 18** ([4, Theorem 4.10])**.** *For every symmetric function $f : \{-1,1\}^n \to \{-1,1\}$, we have $\mathsf{Q}(f) = \Theta(\sqrt{(n-\Gamma(f))n})$.*

The upper bound follows from a quantum algorithm that exactly counts the Hamming weight $|x|$ of the input if $|x| \leq t$ or $|x| \geq n-t$ for $t = \lceil (n-\Gamma(f))/2 \rceil$, and that otherwise learns $|x|$ is in the interval $[t+1, n-t-1]$ (which is an interval around $n/2$ where $f(x)$ is constant). By the definition of $\Gamma(f)$, this information about $|x|$ suffices to compute $f(x)$. In Section 3 we use this observation to give an efficient quantum communication protocol for a two-party function $f \circ G$.

We will need a unitary protocol that allows Alice and Bob to implement an approximate reflection about the n-dimensional maximally entangled state

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i \in \{0,1\}^{\log n}} |i\rangle|i\rangle.$$

Ideally, such a reflection would map $|\psi\rangle$ to itself, and put a minus sign in front of all states orthogonal to $|\psi\rangle$. Doing this perfectly would requires $O(\log n)$ qubits of communication. Fortunately we can derive a cheaper protocol from a test that Aharonov et al. [2, Theorem 1] designed, which uses $O(\log(1/\varepsilon))$ qubits of communication and checks whether a given bipartite state equals $|\psi\rangle$, with one-sided error probability ε. By the usual trick of running this protocol, applying a Z-gate to the answer qubit, and then reversing the protocol, we can implement the desired reflection approximately.[2] A bit more precisely:

▶ **Theorem 19.** *Let $R_\psi = 2|\psi\rangle\langle\psi| - I$ be the reflection about the maximally entangled state shared between Alice and Bob. There exists a protocol that uses $O(\log(1/\varepsilon))$ qubits of communication and that implements a unitary R_ψ^ε such that $\left\| R_\psi^\varepsilon - R_\psi \right\| \leq \varepsilon$ and $R_\psi^\varepsilon|\psi\rangle = |\psi\rangle$.*

We use $\mathsf{UPP}^{cc}(F)$ to denote unbounded-error quantum communication complexity of two-party function F. It is folklore (see for example [15]) that the unbounded-error quantum communication complexity[3] of F equals its classical counterpart up to a factor of at most 2

[2] Possibly with some auxiliary qubits on Alice and Bob's side which start in $|0\rangle$ and end in $|0\rangle$, except in a part of the final state that has norm at most ε.

[3] The unbounded-error model does not allow shared randomness or prior shared entanglement (which yields shared randomness by measuring) between Alice and Bob, since any two-party function F would have constant communication complexity in that setting.

so it does not really matter much whether we use UPP^{cc} for classical unbounded-error communication complexity (as it is commonly used) or for quantum unbounded-error complexity. Crucially, for both the complexity of IP_n is linear in n:

▶ **Theorem 20** ([10]). *Let n be a positive integer. Then $\mathsf{UPP}^{cc}(\mathsf{IP}_n) = \Omega(n)$.*

3 No log-factor needed for symmetric functions

We present a version of quantum amplitude amplification that still works if the reflections involved are not perfectly implemented. In particular, the usual reflection about the uniform superposition will be replaced in the communication setting by an imperfect reflection about the n-dimensional maximally entangled state, based on the communication-efficient protocol of Aharonov et al. [2, Theorem 1] for testing whether Alice and Bob share that state. This allows us to avoid the $\log n$ factor that would be incurred if we instead used a BCW-style distributed implementation of standard amplitude amplification, with $O(\log n)$ qubits of communication to implement each query. Our main technical contribution for proving Theorem 3 is the following general theorem, which allows us to search among a sequence of two-party instances $(X_1, Y_1), \ldots, (X_n, Y_n)$ for an index $i \in [n]$ where $G(X_i, Y_i) = -1$, for any two-party function G.

▶ **Theorem 21.** *Let $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$ be a two-party function, $X = (X_1, \ldots, X_n) \in \{-1,1\}^{nj}$ and $Y = (Y_1, \ldots, Y_n) \in \{-1,1\}^{nk}$. Define $z = (G(X_1, Y_1), \ldots, G(X_n, Y_n)) \in \{-1,1\}^n$. Assume Alice and Bob start with $\lceil \log n \rceil$ shared EPR-pairs.*

- *There exists a quantum protocol using $O(\sqrt{n}\, \mathsf{Q}_E^{cc}(G))$ qubits of communication that finds (with success probability ≥ 0.99) an $i \in [n]$ such that $z_i = -1$ if such an i exists, and says "no" with probability 1 if no such i exists.*
- *If the number of -1s in z is within a factor of 2 from a known integer t, then the communication can be reduced to $O(\sqrt{n/t}\, \mathsf{Q}_E^{cc}(G))$ qubits.*

We prove Theorem 21 in Appendix A. Consider a symmetric Boolean function $f : \{-1,1\}^n \to \{-1,1\}$. As we explained in Section 2.3, there is an integer $t = \lceil (n - \Gamma(f))/2 \rceil$ such that we can compute f if we learn the Hamming weight $|z|$ of the input $z \in \{-1,1\}^n$ or learn that $|z| \in [t+1, n-t-1]$. The bounded-error quantum query complexity is $\mathsf{Q}(f) = \Theta(\sqrt{tn})$ (Theorem 18). We now prove Theorem 3 assuming Theorem 21.

For a given two-party function $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$, we have an induced two-party function $F : \{-1,1\}^{nj} \times \{-1,1\}^{nk} \to \{-1,1\}$ defined as $F(X_1, \ldots, X_n, Y_1, \ldots, Y_n) = f(G(X_1, Y_1), \ldots, G(X_n, Y_n))$. Define

$$z = (G(X_1, Y_1), \ldots, G(X_n, Y_n)) \in \{-1,1\}^n .$$

Then $F(X, Y) = f(z)$ only depends on the number of -1s in z. The following theorem allows us to count this number using $O(\mathsf{Q}(f)\, \mathsf{Q}_E^{cc}(G))$ qubits of communication.

▶ **Theorem 22.** *For every t between 1 and $n/2$, there exists a quantum protocol that starts from $O(t \log n)$ EPR-pairs, communicates $O(\sqrt{tn}\, \mathsf{Q}_E^{cc}(G))$ qubits, and tells us $|z|$ or tells us that $|z| > t$, with error probability $\leq 1/8$.*

Proof. Abbreviate $q = \mathsf{Q}_E^{cc}(G)$. Our protocol has two parts: the first filters out the case $|z| \geq 2t$, while the second finds all solutions if $|z| < 2t$.

Part 1. First Alice and Bob decide between the case (1) $|z| \geq 2t$ and (2) $|z| \leq t$ (even though $|z|$ might also lie in $\{t+1, \ldots, 2t-1\}$) using $O(\sqrt{nq})$ qubits of communication, as follows. They use shared randomness to choose a uniformly random subset $S \subseteq [n]$ of $\lceil n/(2t) \rceil$ elements. Let E be the event that $z_i = -1$ for at least one $i \in S$. By standard calculations there exist $p_1, p_2 \in [0,1]$ with $p_1 = p_2 + \Omega(1)$ such that $\Pr[E] \geq p_1$ in case (1) and $\Pr[E] \leq p_2$ in case (2). Alice and Bob use the distributed-search protocol from the first bullet of Theorem 21 to decide E, with $O(\sqrt{|S|}\, q) = O(\sqrt{n}\, q)$ qubits of communication (plus a negligible $O(\log n)$ EPR-pairs) and error probability much smaller than $p_1 - p_2$. By repeating this a sufficiently large constant number of times and seeing whether the fraction of successes was larger or smaller than $(p_1 + p_2)/2$, they can distinguish between cases (1) and (2) with success probability $\geq 15/16$. If they conclude they're in case (1) then they output "$|z| > t$" and otherwise they proceed to the second part of the protocol.

Note that if $|z| \in \{t+1, \ldots, 2t-1\}$ (the "grey zone" in between cases (1) and (2)), then we can't give high-probability guarantees for one output or the other, but concluding (1) leads to the correct output "$|z| > t$" in this case, while concluding (2) means the protocol proceeds to Part 2. So either course of action is fine if $|z| \in \{t+1, \ldots, 2t-1\}$.

By Newman's theorem [18] the shared randomness used for choosing S can be replaced by $O(\log n)$ bits of private randomness on Alice's part, which she can send to Bob in her first message, so Part 1 communicates $O(\sqrt{n}\, q)$ qubits in total.

Part 2. We condition on Part 1 successfully filtering out case (1), so from now on assume $|z| < 2t$. Our goal in this second part of the protocol is to find all indices i such that $z_i = -1$ (we call such i "solutions"), with probability $\geq 15/16$, using $O(\sqrt{tn}\, q)$ qubits of communication. This will imply that the overall protocol is correct with probability $1 - 1/16 - 1/16 = 7/8$, and uses $O(\sqrt{tn}\, q)$ qubits of communication in total. For an integer $k \geq 1$, consider the following protocol P_k.

Algorithm 1 Protocol P_k.

```
Input: An integer $k \geq 1$
repeat
    1. Run the protocol from the last bullet of Theorem 21 with $t = 2^{k-1}$.
       (suppressing some constant factors, assume for simplicity that this uses
       $\sqrt{n/2^k}\, q$ qubits of communication, $\log n$ shared EPR-pairs at the start, and has
       probability $\geq 1/100$ to find a solution if the actual number of solutions is in
       $[t/2, 2t]$).
    2. Alice measures and gets outcome $i \in [n]$ and Bob measures and gets outcome
       $j \in [n]$, respectively.
    3. Alice sends $i$ to Bob, Bob sends $j$ to Alice.
    4. If $i = j$ then they verify that $G(X_i, Y_i) = -1$ by one run of the protocol for $G$,
       and if so then they replace $X_i, Y_i$ by some pre-agreed inputs $X_i', Y_i'$,
       respectively, such that $G(X_i', Y_i') = 1$ (this reduces the number of $-1$s in $z$ by 1)
until $200\sqrt{2^k n}\, q$ qubits have been sent;
```

▷ Claim 23. Suppose $|z| \in [2^{k-1}, 2^k)$. Then protocol P_k uses $O(\sqrt{2^k n}\, q)$ qubits of communication, assumes $O(2^k \log n)$ EPR-pairs at the start of the protocol, and finds at least $|z| - 2^{k-1} + 1$ solutions, except with probability $\leq 1/2$.

Proof. The upper bound on the communication is obvious from the stopping criterion of P_k.

As long as the remaining number of solutions is $\geq 2^{k-1}$, each run of the protocol has probability $\geq 1/100$ to find another solution. Hence the expected number of runs of the protocol of Theorem 21 to find at least $|z| - 2^{k-1} + 1$ solutions, is $\leq 100(|z| - 2^{k-1} + 1)$. By Markov's inequality, the probability that we haven't yet found $|z| - 2^{k-1} + 1$ solutions after $\leq 200(|z| - 2^{k-1} + 1) \leq 100 \cdot 2^k$ runs, is $\leq 1/2$. The communication cost of so many runs is $100 \cdot 2^k(\sqrt{n/2^k}\,q + \log n) \leq 200\sqrt{2^k n}\,q$ qubits. Hence by the time that the number of qubits of the stopping criterion have been communicated, we have probability $\geq 1/2$ of having found at least $|z| - 2^{k-1} + 1$ solutions. The assumed number of EPR-pairs at the start is $\log n$ per run, so $O(2^k \log n)$ in total. ◁

Note that if we start with a number of solutions $|z| \in [2^{k-1}, 2^k)$, and P_k succeeds in finding at least $|z| - 2^{k-1} + 1$ new solutions, then afterwards we have $< 2^{k-1}$ solutions left. The following protocol runs these P_k in sequence, pushing down the remaining number of solutions to 0.

Algorithm 2 Protocol P.

```
for $k = \lceil \log_2(2t) \rceil$ downto 1 do
    1. Run $P_k$ a total of $r_k = \lceil \log_2(2t) \rceil - k + 5$ times (replacing all $-1$s found by
       $+1s$ in $z$).
    2. Output the total number of solutions found.
end
```

▷ Claim 24. If $|z| < 2t$ then protocol $\mathcal{P}$ uses $O(\sqrt{tn}\,q)$ qubits of communication, assumes $O(t \log n)$ EPR-pairs at the start of the protocol, and outputs $|z|$, except with probability $\leq 1/16$.

Proof. First, by Claim 23, the total number of qubits communicated is

$$\sum_{k=1}^{\lceil \log_2(2t) \rceil} r_k \cdot O(\sqrt{2^k n}\,q) = O(\sqrt{tn}\,q) \cdot \sum_{\ell=0}^{\lceil \log_2(2t) \rceil - 1} (\ell + 5)/\sqrt{2^\ell} = O(\sqrt{tn}\,q),$$

where we used a variable substitution $k = \lceil \log_2(2t) \rceil - \ell$. Second, the number of EPR-pairs we're starting from is

$$\sum_{k=1}^{\lceil \log_2(2t) \rceil} r_k \cdot O(2^k \log n) = O(t \log n) \cdot \sum_{\ell=0}^{\lceil \log_2(2t) \rceil - 1} (\ell + 5)/2^\ell = O(t \log n).$$

Third, by Claim 23 and the fact that we are performing r_k repetitions of P_k, if the kth round of $\mathcal{P}$ starts with a remaining number of solutions that is in the interval $[2^{k-1}, 2^k)$ then that round ends with $< 2^{k-1}$ remaining solutions, except with probability at most $1/2^{r_k}$. By the union bound, the probability that any one of the $\lceil \log_2(2t) \rceil$ rounds does not succeed at this, is at most

$$\sum_{k=1}^{\lceil \log_2(2t) \rceil} \frac{1}{2^{r_k}} = \sum_{\ell=0}^{\lceil \log_2(2t) \rceil - 1} \frac{1}{2^{\ell+5}} \leq \frac{1}{16}.$$

Since $2^{\lceil \log_2(2t) \rceil} \geq 2t$ and we start with $|z| < 2t$, if each round succeeds, then by the end of $\mathcal{P}$ there are no remaining solutions left. Thus, the protocol $\mathcal{P}$ finds all solutions and learns $|z|$ with probability at least $15/16$. ◁

Part 1 and Part 2 each have error probability $\leq 1/16$, so by the union bound the protocol succeeds except with probability $1/8$. If $|z| \geq 2t$ then Part 1 outputs the correct answer "$|z| > t$"; if $|z| \leq t$ then all solutions (and hence $|z|$) are found by Part 2; and if $|z| \in \{t+1, \ldots, 2t-1\}$ then either Part 1 already outputs the correct answer "$|z| > t$" or the protocol proceeds to Part 2 which then finds all solutions. ◀

We can use the above theorem twice: once to count the number of -1s in z (up to t) and once to count the number of 1s in z (up to t). This uses $O(\sqrt{tn}\,\mathsf{Q}_E^{cc}(G)) = O(\mathsf{Q}(f)\,\mathsf{Q}_E^{cc}(G))$ qubits of communication, assumes $O(t \log n)$ shared EPR-pairs at the start of the protocol, and gives us enough information about $|z|$ to compute $f(z) = F(X, Y)$. This concludes the proof of Theorem 3 from the introduction, restated below.

▶ **Theorem 25** (Restatement of Theorem 3). *For every symmetric Boolean function* $f : \{-1,1\}^n \to \{-1,1\}$ *and two-party function* $G : \{-1,1\}^j \times \{-1,1\}^k \to \{0,1\}$*, we have* $\mathsf{Q}^{cc,*}(f \circ G) = O(\mathsf{Q}(f)\mathsf{Q}_E^{cc}(G))$.

If $\mathsf{Q}(f) = \Theta(\sqrt{tn})$, then our protocol in the proof of Theorem 3 assumes a shared state of $O(t \log n)$ EPR-pairs at the start. We remark that for the special case where $G = \mathsf{AND}_2$, our upper bound matches the lower bound proved by Razborov [22], except for symmetric functions f where the first switch of function value happens at Hamming weights very close to n. In particular, if $f = \mathsf{AND}_n$ and $G = \mathsf{AND}_2$, then $\mathsf{Q}^{cc}(f \circ G) = 1$ but $\mathsf{Q}(f) = \Theta(\sqrt{n})$.

4 Necessity of the log-factor overhead in the BCW simulation

In this section we prove Theorem 5. We exhibit a function $f : \{-1,1\}^{2n^2} \to \{-1,1\}$ for which $\mathsf{Q}(f) = O(n)$ and $\mathsf{UPP}(f \circ \square) = \Omega(n \log n)$ for $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$.

The proofs of Theorem 5 and Theorem 6 each involve proving a query complexity upper bound and a communication complexity lower bound. The proofs of the query complexity upper bounds are along similar lines and follow from Theorem 29 and Corollary 30 (see Section 4.1). The proofs of the communication complexity lower bounds each involve a reduction from a problem whose communication complexity is easier to analyze (see Lemma 31 in Section 4.2). We complete the proof of Theorem 5 in Section 4.2.1. See the full version of our paper [8] for a proof of Theorem 6.

4.1 Quantum query complexity upper bound

For total functions f, g, let $f \circ g$ denote the standard composition of the functions f and g. We also require the following notion of composition of a total function f with a partial function g.

▶ **Definition 26** (Composition with partial functions). *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be a total function and let* $g : \{-1,1\}^m \to \{-1,1,\star\}$ *be a partial function. Let* $f \,\widetilde{\circ}\, g : \{-1,1\}^{nm} \to \{-1,1\}$ *denote the total function that is defined as follows on input* $(X_1, \ldots, X_n) \in \{-1,1\}^{nm}$*, where* $X_i \in \{-1,1\}^m$ *for all* $i \in [n]$.

$$f \,\widetilde{\circ}\, g(X_1, \ldots, X_n) = \begin{cases} f(g(X_1), \ldots, g(X_n)) & \textit{if } g(X_i) \in \{-1,1\} \textit{ for all } i \in [n], \\ -1 & \textit{otherwise.} \end{cases}$$

That is, we use $f \,\widetilde{\circ}\, g$ to denote the total function that equals $f \circ g$ on inputs when each copy of g outputs a value in $\{-1,1\}$, and equals -1 otherwise.

Recall that we index coordinates of n-bit strings by integers in $[n]$, and also interchangeably by strings in $\{-1,1\}^{\log n}$ via the natural correspondence. For $x \in \{-1,1\}^n$, let $-x \in \{-1,1\}^n$ be defined as $(-x)_i = -x_i$ for all $i \in [n]$. We use the notation $\pm x$ to denote the set $\{x, -x\}$.

▶ **Definition 27** (Hadamard Codewords). *For every positive integer n and $s \in \{-1,1\}^{\log n}$, let $H(s) \in \{-1,1\}^n$ be defined as $(H(s))_t = \prod_{i:s_i=-1} t_i$ for all $t \in \{-1,1\}^{\log n}$. If $x \in \{-1,1\}^n$ is such that $x = H(s)$ for some $s \in \{-1,1\}^{\log n}$, we say x is a Hadamard codeword corresponding to s.*

That is, for every $s \in \{-1,1\}^{\log n}$, there is an n-bit Hadamard codeword corresponding to s. This represents the enumeration of all parities of s.

We now define how to encode a two-party total function G on $(\log j + \log k)$ input bits to a partial function h_G on $(j+k)$ input bits, using Hadamard encoding.

▶ **Definition 28** (Hadamardization of functions). *Let $j, k \geq 1$ be powers of 2, and let $G : \{-1,1\}^{\log j} \times \{-1,1\}^{\log k} \to \{-1,1\}$ be a function. Define a partial function $h_G : \{-1,1\}^{j+k} \to \{-1,1,\star\}$ by*

$$h_G(x,y) = \begin{cases} G(s,t) & \text{if } x \in \pm H(s), y \in \pm H(t) \text{ for some } s \in \{-1,1\}^{\log j}, t \in \{-1,1\}^{\log k} \\ \star & \text{otherwise.} \end{cases}$$

We next prove the following theorem. (See Figure 2 for a visual description of h_G.)

▶ **Theorem 29.** *Let $G : \{-1,1\}^{\log j} \times \{-1,1\}^{\log k} \to \{-1,1\}$ and $r : \{-1,1\}^n \to \{-1,1\}$. Then the quantum query complexity of the function $r \mathbin{\tilde{\circ}} h_G : \{-1,1\}^{n(j+k)} \to \{-1,1\}$ is given by $\mathsf{Q}(r \mathbin{\tilde{\circ}} h_G) = O(n + \sqrt{n(j+k)})$.*

Proof. Recall from Definition 26 that the function $r \mathbin{\tilde{\circ}} h_G : \{-1,1\}^{n(j+k)} \to \{-1,1\}$ is defined as $r \mathbin{\tilde{\circ}} h_G((X_1,Y_1),\ldots,(X_n,Y_n)) = r \circ h_G((X_1,Y_1),\ldots,(X_n,Y_n))$ if $h_G((X_i,Y_i)) \in \{-1,1\}$ for all $i \in [n]$, and -1 otherwise.

Quantum query algorithm

View inputs to $r \mathbin{\tilde{\circ}} h_G$ as $(X_1, Y_1, \ldots, X_n, Y_n)$, where $X_i \in \{-1,1\}^j$ for all $i \in [n]$ and $Y_i \in \{-1,1\}^k$ for all $i \in [n]$. We give a quantum algorithm and its analysis below.

1. Run $2n$ instances of the Bernstein-Vazirani algorithm: 1 instance on each X_i and 1 instance on each Y_i, to obtain $2n$ strings $x_1,\ldots,x_n,y_1,\ldots,y_n$, where each x_i is a $(\log j)$-bit string and each y_i is a $(\log k)$-bit string.
2. For each X_i and Y_i, query $(X_i)_{1^{\log j}}$ and $(Y_i)_{1^{\log k}}$ to obtain bits $b_i, c_i \in \{-1,1\}$ for all $i \in [n]$.
3. Run Grover's search [11, 6] to check equality of the following two $(nj+nk)$-bit strings: $(b_1H(x_1),\ldots,b_nH(x_n),c_1H(y_1),\ldots,c_nH(y_n))$ and $(X_1,\ldots,X_n,Y_1,\ldots,Y_n)$.
4. If the step above outputs that the strings are equal, then output $r(G(x_1,y_1),\ldots,G(x_n,y_n))$. Else, output -1.

Analysis of the algorithm

- If the input is indeed of the form $(X_1,Y_1),\ldots,(X_n,Y_n)$ where each $X_i \in \pm H(x_i)$ and $Y_i \in \pm H(y_i)$ for some $x_i \in \{-1,1\}^{\log j}$ and $y_i \in \{-1,1\}^{\log k}$, then Step 1 outputs the correct strings $x_1,\ldots,x_n,y_1,\ldots,y_n$ with probability 1 by the properties of the Bernstein-Vazirani algorithm. Step 2 then implies that $X_i = b_iH(x_i)$ and $Y_i = c_iH(y_i)$ for all $i \in [n]$.

Next, Step 3 outputs that the strings are equal with probability 1 (since the strings whose equality are to be checked are equal). Hence the algorithm is correct with probability 1 in this case, since $(r \mathbin{\tilde{\circ}} h_G)(X_1, Y_1, \ldots, X_n, Y_n) = r(G(x_1, y_1), \ldots, G(x_n, y_n))$.

- If the input is such that there exists an index $i \in [n]$ for which $X_i \notin \pm H(x_i)$ for every $x_i \in \{-1,1\}^{\log j}$ or $Y_i \notin \pm H(y_i)$ for every $y_i \in \{-1,1\}^{\log k}$, then the two strings for which equality is to be checked in the Step 3 are not equal. Grover's search catches a discrepancy with probability at least $2/3$. Hence, the algorithm outputs -1 (as does $r \mathbin{\tilde{\circ}} h_G$), and is correct with probability at least $2/3$ in this case.

Cost of the algorithm

Step 1 accounts for $2n$ quantum queries. Step 2 accounts for $2n$ quantum queries. Step 3 accounts for $O(\sqrt{n(j+k)})$ quantum queries. Thus, $\mathsf{Q}(r \mathbin{\tilde{\circ}} h_G) = O(n + \sqrt{n(j+k)})$. ◀

As a corollary to Theorem 29, we obtain the following on instantiating $j = k = n$ and r as a Boolean function with quantum query complexity $\Theta(n)$.

▶ **Corollary 30.** *Let $G : \{-1,1\}^{\log n} \times \{-1,1\}^{\log n} \to \{-1,1\}$ be a non-constant function and let $r : \{-1,1\}^n \to \{-1,1\}$ be a total function with $\mathsf{Q}(r) = \Theta(n)$. Then the quantum query complexity of the total function $r \mathbin{\tilde{\circ}} h_G : \{-1,1\}^{2n^2} \to \{-1,1\}$ is $\mathsf{Q}(r \mathbin{\tilde{\circ}} h_G) = \Theta(n)$.*

Proof. The upper bound $\mathsf{Q}(r \mathbin{\tilde{\circ}} h_G) = O(n)$ follows by plugging in parameters in Theorem 29.

For the lower bound, we show that $\mathsf{Q}(r \mathbin{\tilde{\circ}} h_G) \geq \mathsf{Q}(r)$. Since G is non-constant, there exist $x_1, y_1, x_2, y_2 \in \{-1,1\}^{\log n}$ such that $G(x_1, y_1) = -1$ and $G(x_2, y_2) = 1$. Let $X_1 = H(x_1), Y_1 = H(y_1)$, $X_2 = H(x_2)$ and $X_2 = H(y_2)$. Consider $r \mathbin{\tilde{\circ}} h_G$ only restricted to inputs where the inputs to each copy of h_G are either (X_1, Y_1) or (X_2, Y_2). Under this restriction, $r \mathbin{\tilde{\circ}} h_G : \{-1,1\}^{2n^2} \to \{-1,1\}$ is the same as $r : \{-1,1\}^n \to \{-1,1\}$. Thus $\mathsf{Q}(r \mathbin{\tilde{\circ}} h_G) \geq \mathsf{Q}(r) = \Omega(n)$. ◀

4.2 On the tightness of the BCW simulation

In this section we first state a communication lower bound (under some model) on $(r \mathbin{\tilde{\circ}} h_G) \circ \square$ in terms of the communication complexity of $r \circ G$ (in the same model of communication). We state the lemma below (Lemma 31) for the case where the models under consideration are the bounded-error and unbounded-error quantum models, since these are the models of interest to us.

▶ **Lemma 31.** *Let $r : \{-1,1\}^n \to \{-1,1\}$, $G : \{-1,1\}^{\log j} \times \{-1,1\}^{\log k} \to \{-1,1\}$, $\square \in \{\mathsf{AND}_2, \mathsf{XOR}_2\}$ and $CC \in \{\mathsf{Q}^{cc,*}, \mathsf{UPP}^{cc}\}$. Then $CC((r \mathbin{\tilde{\circ}} h_G) \circ \square) \geq CC(r \circ G)$.*

The proof of this lemma follows by a simple reduction. We refer the reader to the full version [8] for a formal proof.

4.2.1 Proof of Theorem 5

The total function $f : \{-1,1\}^{2n^2} \to \{-1,1\}$ that we use to prove Theorem 5 is $f = r \mathbin{\tilde{\circ}} h_G$, where $r = \mathsf{PARITY}_n$ and $G = \mathsf{IP}_{\log n}$. The following claim shows that f is transitive.

▷ Claim 32. Let $n > 0$ be a power of 2. Let $r = \mathsf{PARITY}_n : \{-1,1\}^n \to \{-1,1\}$ and $G = \mathsf{IP}_{\log n} : \{-1,1\}^{\log n} \times \{-1,1\}^{\log n} \to \{-1,1\}$. The function $f = r \mathbin{\tilde{\circ}} h_G : \{-1,1\}^{2n^2} \to \{-1,1\}$ is transitive.

Proof. We first show that $h_G : \{-1,1\}^{2n} \to \{-1,1\}$ is transitive. We next observe that $s \tilde{\circ} t$ is transitive whenever s is symmetric and t is transitive. The theorem then follows since PARITY_n is symmetric.

Towards showing transitivity of h_G, let $\pi \in S_{2n}$, and $(\sigma_\ell, \sigma_\ell) \in S_{2n}$ for $\ell \in \{-1,1\}^{\log n}$ be defined as follows. (Here $\sigma_\ell \in S_n$; the first copy acts on the first n coordinates, and the second copy acts on the next n coordinates.)

$$\pi(k) = \begin{cases} k+n & k \leq n \\ k-n & k > n. \end{cases}$$

That is, on every string $(x,y) \in \{-1,1\}^{2n}$, the permutation π maps (x,y) to (y,x).

For every $\ell \in \{-1,1\}^{\log n}$, the permutation $\sigma_\ell \in S_n$ is defined as

$$\sigma_\ell(i) = i \oplus \ell, \tag{2}$$

where $i \oplus \ell$ denotes the bitwise XOR of the strings i and ℓ. That is, for every input $(x,y) \in \{-1,1\}^{2n}$ and every $k \in \{-1,1\}^{\log n}$, the input bit x_k is mapped to $x_{k \oplus \ell}$ and y_k is mapped to $y_{k \oplus \ell}$.

For every $(x,y) \in \{-1,1\}^{2n}$ and $i,j \in \{-1,1\}^{\log n}$, the permutation $\sigma_{i \oplus j}(x,y)$ swaps x_i and x_j, and also swaps y_i and y_j. If for $i,j \in \{-1,1\}^{\log n}$, our task was to swap the i'th index of the first n variables with the j'th index of the second n variables, then the permutation $\sigma_{i \oplus j} \circ \pi$ does the job. That is, for every $(x,y) \in \{-1,1\}^{2n}$ and $i,j \in \{-1,1\}^{\log n}$, the permutation $\sigma_{i \oplus j} \circ \pi$ maps x_i to y_j. Thus the set of permutations $\{\pi, \{\sigma_\ell : \ell \in \{-1,1\}^{\log n}\}\}$ acts transitively on S_{2n}.

Now we show that for all $x,y \in \{-1,1\}^{2n}$ and all $\ell \in \{-1,1\}^{\log n}$, we have $h_G(\sigma_\ell(x), \sigma_\ell(y)) = h_G(x,y)$. Fix $\ell \in \{-1,1\}^{\log n}$.

- If $x \in \pm H(s)$ and $y \in \pm H(t)$ are Hadamard codewords, then $x_k = \langle k, s \rangle$ and $y_k = \langle k, t \rangle$ for all $k \in \{-1,1\}^{\log n}$, and $G(x,y) = \langle s,t \rangle$. Thus, for every $k \in \{-1,1\}^{\log n}$ we have $\sigma_\ell(x_k) = x_{k \oplus \ell} = \langle k \oplus \ell, s \rangle = \langle \ell, s \rangle \cdot \langle k, s \rangle$. Hence $\sigma_\ell(x) \in \pm H(s)$ (since $\langle \ell, s \rangle$ does not depend on k, and takes value either 1 or -1). Similarly, $\sigma_\ell(y) \in \pm H(t)$. Thus $h_G(\sigma_\ell(x,y)) = h_G(x,y)$.
- If x (y, respectively) is not a Hadamard codeword, then a similar argument shows that for all $\ell \in [n]$, $\sigma_\ell(x)$ ($\sigma_\ell(y)$, respectively) is also not a Hadamard codeword.

Using the fact that $\langle s,t \rangle = \langle t,s \rangle$ for every $s,t \in \{-1,1\}^{\log n}$, one may verify that $h_G(\pi(x,y)) = h_G(x,y)$ for all $x,y \in \{-1,1\}^{2n}$.

Along with the observation that PARITY_n is a symmetric function, we have that $f = r \tilde{\circ} h_G : \{-1,1\}^{2n^2} \to \{-1,1\}$ is transitive under the following permutations:

- S_n acting on the inputs of PARITY_n, and
- The group generated by $\{\pi\} \cup \{(\sigma_\ell, \sigma_\ell) : \ell \in [n]\}$ acting independently on the inputs of each copy of h_G, where σ_ℓ is as in Equation (2). ◁

Proof of Theorem 5. Let $n > 0$ be a power of 2. Let $r = \mathsf{PARITY}_n : \{-1,1\}^n \to \{-1,1\}$ and $G = \mathsf{IP}_{\log n} : \{-1,1\}^{\log n} \times \{-1,1\}^{\log n} \to \{-1,1\}$. Let $f = r \tilde{\circ} h_G : \{-1,1\}^{2n^2} \to \{-1,1\}$. By Claim 32, f is transitive. By Corollary 30 we have $\mathsf{Q}(f) = \Theta(n)$. For the communication lower bound we have

$$\begin{aligned} \mathsf{UPP}^{cc}(f \circ \square) &= \mathsf{UPP}^{cc}((r \tilde{\circ} h_G) \circ \square) \\ &\geq \mathsf{UPP}^{cc}(\mathsf{PARITY}_n \circ \mathsf{IP}_{\log n}) && \text{by Lemma 31} \\ &= \mathsf{UPP}^{cc}(\mathsf{IP}_{n \log n}) && \text{Observation 15} \\ &= \Omega(n \log n). && \text{by Theorem 20} \end{aligned}$$

◀

References

1 Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005. Earlier version in FOCS'03. quant-ph/0303041.

2 Dorit Aharonov, Aram W. Harrow, Zeph Landau, Daniel Nagaj, Mario Szegedy, and Umesh V. Vazirani. Local tests of global entanglement and a counterexample to the generalized area law. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 246–255, 2014. doi:10.1109/FOCS.2014.34.

3 Andris Ambainis and Ronald de Wolf. How low can approximate degree and quantum query complexity be for total Boolean functions? *Computational Complexity*, 23(2):305–322, 2014. Earlier version in CCC'13.

4 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. quant-ph/9802049.

5 Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Earlier version in STOC'93.

6 Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. American Mathematical Society, 2002. arXiv:quant-ph/0005055.

7 Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 63–68, 1998. doi:10.1145/276698.276713.

8 Sourav Chakraborty, Arkadev Chattopadhyay, Peter Høyer, Nikhil S. Mande, Manaswi Paraashar, and Ronald de Wolf. Symmetry and quantum query-to-communication simulation. *CoRR*, abs/2012.05233, 2020. arXiv:2012.05233.

9 Sourav Chakraborty, Arkadev Chattopadhyay, Nikhil S. Mande, and Manaswi Paraashar. Quantum query-to-communication simulation needs a logarithmic overhead. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 32:1–32:15, 2020. doi:10.4230/LIPIcs.CCC.2020.32.

10 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and Systems Sciences*, 65(4):612–625, 2002. doi:10.1016/S0022-0000(02)00019-3.

11 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219, 1996.

12 Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Sign rank vs discrepancy. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 18:1–18:14, 2020. doi:10.4230/LIPIcs.CCC.2020.18.

13 Peter Høyer and Ronald de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 299–310, 2002.

14 Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2719 of *Lecture Notes in Computer Science*, pages 291–299. Springer, 2003. quant-ph/0304052.

15 Kazuo Iwama, Harumichi Nishimura, Rudy Raymond, and Shigeru Yamashita. Unbounded-error one-way classical and quantum communication complexity. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 110–121. Springer, 2007.

16 Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1997.

17 Troy Lee and Shengyu Zhang. Composition theorems in communication complexity. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming, (ICALP)*, pages 475–489, 2010. doi:10.1007/978-3-642-14165-2_41.

18 Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991.

19 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

20 Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 468–474, 1992. doi:10.1145/129712.129758.

21 Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *Journal of Computer and System Sciences*, 33(1):106–123, 1986.

22 Alexander Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Sciences, mathematics*, 67(1):159–176, 2003. quant-ph/0204025.

23 Ronald de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002.

24 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

25 Andrew Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science (FOCS)*, pages 352–361. IEEE, 1993.

A Noisy amplitude amplification and a new distributed-search protocol

In this section we prove Theorem 21, restated below.

▶ **Theorem 33** (Restatement of Theorem 21). *Let $G : \{-1,1\}^j \times \{-1,1\}^k \to \{-1,1\}$ be a two-party function, $X = (X_1,\ldots,X_n) \in \{-1,1\}^{nj}$ and $Y = (Y_1,\ldots,Y_n) \in \{-1,1\}^{nk}$. Define $z = (G(X_1,Y_1),\ldots,G(X_n,Y_n)) \in \{-1,1\}^n$. Assume Alice and Bob start with $\lceil \log n \rceil$ shared EPR-pairs.*

- *There exists a quantum protocol using $O(\sqrt{n}\,\mathsf{Q}^{cc}_E(G))$ qubits of communication that finds (with success probability ≥ 0.99) an $i \in [n]$ such that $z_i = -1$ if such an i exists, and says "no" with probability 1 if no such i exists.*
- *If the number of -1s in z is within a factor of 2 from a known integer t, then the communication can be reduced to $O(\sqrt{n/t}\,\mathsf{Q}^{cc}_E(G))$ qubits.*

▶ Remark 34. The $\log n$ shared EPR-pairs that we assume Alice and Bob share at the start could also be established by means of $\log n$ qubits of communication at the start of the protocol. For the result in the first bullet, this additional communication does not change the asymptotic bound. For the result of the second bullet, if $t \leq n\mathsf{Q}^{cc}_E(G)^2/(\log n)^2$, then this additional communication does not change the asymptotic bound either. However, if $t = \omega(n/(\log n)^2)$ and $\mathsf{Q}^{cc}_E(G) = O(1)$ then the quantum communication $O(\sqrt{n/t}\,\mathsf{Q}^{cc}_E(G))$ is $o(\log n)$ and establishing the $\log n$ EPR-pairs by means of a first message makes a difference.

As a corollary, we obtain a new $O(\sqrt{n})$-qubit protocol for the distributed search problem composed with $G = \mathsf{AND}_2$ (whose decision version is the Set-Disjointness problem).

A.1 Amplitude amplification with perfect reflections

We first describe basic amplitude amplification in a slightly unusual recursive manner, similar to [14]. We are dealing with a search problem where some set $\mathcal{G}$ of basis states are deemed "good" and the other basis states are deemed "bad." Let $P_{\mathcal{G}} = \sum_{g\in\mathcal{G}} |g\rangle\langle g|$ be the projector onto the span of the good basis states, and $O_{\mathcal{G}} = I - 2P_{\mathcal{G}}$ be the reflection that puts a "$-$" in front of the good basis states: $O_{\mathcal{G}}|g\rangle = -|g\rangle$ for all basis states $g \in \mathcal{G}$, and $O_{\mathcal{G}}|b\rangle = |b\rangle$ for all basis states $b \notin \mathcal{G}$.

Suppose we have an initial state $|\psi\rangle$ which is a superposition of a good state and a bad state:

$$|\psi\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle,$$

where $|G\rangle = P_{\mathcal{G}}|\psi\rangle / \|P_{\mathcal{G}}|\psi\rangle\|$ and $|B\rangle = (I - P_{\mathcal{G}})|\psi\rangle / \|(I - P_{\mathcal{G}})|\psi\rangle\|$. For example in Grover's algorithm, with a search space of size n containing t solutions, the initial state $|\psi\rangle$ would be the uniform superposition, and its overlap (inner product) with the good subspace spanned by the t "good" (sometimes called "marked") basis states would be $\sin(\theta) = \sqrt{t/n}$.

We'd like to increase the weight of the good state, i.e., move the angle θ closer to $\pi/2$. Let R_ψ denote the reflection about the state $|\psi\rangle$, i.e., $R_\psi|\psi\rangle = |\psi\rangle$ and $R_\psi|\phi\rangle = -|\phi\rangle$ for every $|\phi\rangle$ that is orthogonal to $|\psi\rangle$. Then the algorithm $A_1 = R_\psi \cdot O_{\mathcal{G}}$ is the product of two reflections, which (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$) corresponds to a rotation by an angle 2θ, thus increasing our angle from θ to 3θ. This is the basic amplitude amplification step. It maps

$$|\psi\rangle \mapsto A_1|\psi\rangle = \sin(3\theta)|G\rangle + \cos(3\theta)|B\rangle.$$

We can now repeat this step recursively, defining

$$A_2 = A_1 R_\psi A_1^* \cdot O_{\mathcal{G}} \cdot A_1.$$

Note that $A_1 R_\psi A_1^*$ is a reflection about the state $A_1|\psi\rangle$. Thus A_2 triples the angle between $A_1|\psi\rangle$ and $|B\rangle$, mapping

$$|\psi\rangle \mapsto A_2|\psi\rangle = \sin(9\theta)|G\rangle + \cos(9\theta)|B\rangle.$$

Continuing recursively in this fashion, define the algorithm

$$A_{j+1} = A_j R_\psi A_j^* \cdot O_{\mathcal{G}} \cdot A_j. \tag{3}$$

The last algorithm A_k will map

$$|\psi\rangle \mapsto A_k|\psi\rangle = \sin(3^k\theta)|G\rangle + \cos(3^k\theta)|B\rangle.$$

Hence after k recursive amplitude amplification steps, we have angle $3^k\theta$. Since we want to end up with angle $\approx \pi/2$, if we know θ then we can choose

$$k = \lfloor \log_3(\pi/(2\theta)) \rfloor . \tag{4}$$

This gives us an angle $3^k\theta \in (\pi/6, \pi/2]$, so the final state $A_k|\psi\rangle$ has overlap $\sin(\theta_k) > 1/2$ with the good state $|G\rangle$.

Let C_k denote the "cost" (in whatever measure, for example query complexity, or communication complexity, or circuit size) of algorithm A_k. Looking at its recursive definition (Equation (3)), C_k is 3 times C_{k-1}, plus the cost of R_ψ plus the cost of $O_{\mathcal{G}}$. If we just count applications of $O_{\mathcal{G}}$ ("queries"), considering R_ψ to be free, then $C_{k+1} = 3C_k + 1$. This recursion has the closed form $C_k = \sum_{i=0}^{k-1} 3^i < 3^k$. With the above choice of k we get $C_k = O(1/\theta)$. In the case of Grover's algorithm, where $\theta = \arcsin(\sqrt{t/n}) \approx \sqrt{t/n}$, the cost is $C_k = O(\sqrt{n/t})$.

A.2 Amplitude amplification with imperfect reflections

Now we consider the situation where we do not implement the reflections R_ψ perfectly, but instead implement another unitary R_ψ^ε at operator-norm distance $\left\|R_\psi^\varepsilon - R_\psi\right\| \leq \varepsilon$ from R_ψ, with the additional property that $R_\psi^\varepsilon|\psi\rangle = |\psi\rangle$ (this one-sided error property will be important for the proof). We can control this error ε, but smaller ε will typically correspond to higher cost of R_ψ^ε. The reflection $O_\mathcal{G}$ will still be implemented perfectly below.

We again start with the initial state

$$|\psi\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle.$$

For errors $\varepsilon_1, \ldots, \varepsilon_k$ that we will specify later, recursively define the following algorithms.

$$A_1 = R_\psi^{\varepsilon_1} \cdot O_\mathcal{G} \quad \text{and} \quad A_{j+1} = A_j R_\psi^{\varepsilon_{j+1}} A_j^* \cdot O_\mathcal{G} \cdot A_j.$$

These algorithms will map the initial state as follows:

$$|\psi\rangle \mapsto |\psi_j\rangle = A_j|\psi\rangle = \sin(3^j\theta)|G\rangle + \cos(3^j\theta)|B\rangle + |E_j\rangle, \tag{5}$$

where $|E_j\rangle$ is some unnormalized error state defined by the above equation; its norm η_j quantifies the extent to which we deviate from perfect amplitude amplification. Our goal here is to upper bound this η_j. In order to see how η_j can grow, let us see how $A_j R_\psi^{\varepsilon_{j+1}} A_j^* \cdot O_\mathcal{G}$ acts on $\sin(3^j\theta)|G\rangle + \cos(3^j\theta)|B\rangle$ (we'll take into account the effects of the error term $|E_j\rangle$ later). If $R_\psi^{\varepsilon_{j+1}}$ were equal to R_ψ, then we would have one perfect round of amplitude amplification and obtain $\sin(3^{j+1}\theta)|G\rangle + \cos(3^{j+1}\theta)|B\rangle$; but since $R_\psi^{\varepsilon_{j+1}}$ is only ε_{j+1}-close to R_ψ, additional errors can appear. First we apply $O_\mathcal{G}$, which flips the phase of $|G\rangle$ and hence changes the state to

$$-\sin(3^j\theta)|G\rangle + \cos(3^j\theta)|B\rangle = |\psi_j\rangle - |E_j\rangle - 2\sin(3^j\theta)|G\rangle.$$

Second we apply $V = A_j R_\psi^{\varepsilon_{j+1}} A_j^*$. Let $V' = A_j R_\psi A_j^*$, and note that $V|\psi_j\rangle = V'|\psi_j\rangle = |\psi_j\rangle$ and $\|V' - V\| = \left\|R_\psi - R_\psi^{\varepsilon_{j+1}}\right\| \leq \varepsilon_{j+1}$. The new state is

$$\begin{aligned} V(|\psi_j\rangle - |E_j\rangle - 2\sin(3^j\theta)|G\rangle) &= V'(|\psi_j\rangle - |E_j\rangle - 2\sin(3^j\theta)|G\rangle) + (V'-V)(|E_j\rangle + 2\sin(3^j\theta)|G\rangle) \\ &= V'(-\sin(3^j\theta)|G\rangle + \cos(3^j\theta)|B\rangle) + (V'-V)(|E_j\rangle + 2\sin(3^j\theta)|G\rangle) \\ &= \sin(3^{j+1}\theta)|G\rangle + \cos(3^{j+1}\theta)|B\rangle + (V'-V)(|E_j\rangle + 2\sin(3^j\theta)|G\rangle). \end{aligned}$$

Putting back also the earlier error term $|E_j\rangle$ from Equation (5) (to which the unitary $VO_\mathcal{G}$ is applied as well), it follows that the new error state is

$$\begin{aligned} |E_{j+1}\rangle &= |\psi_{j+1}\rangle - (\sin(3^{j+1}\theta)|G\rangle + \cos(3^{j+1}\theta)|B\rangle) \\ &= VO_\mathcal{G}|E_j\rangle + (V' - V)(|E_j\rangle + 2\sin(3^j\theta)|G\rangle). \end{aligned}$$

Its norm is

$$\begin{aligned} \eta_{j+1} &\leq \|VO_\mathcal{G}|E_j\rangle\| + \left\|(V' - V)(|E_j\rangle + 2\sin(3^j\theta)|G\rangle)\right\| \\ &\leq \eta_j + \varepsilon_{j+1}(\eta_j + 2\sin(3^j\theta)) = (1 + \varepsilon_{j+1})\eta_j + 2\varepsilon_{j+1}\sin(3^j\theta). \end{aligned}$$

Since $\eta_0 = 0$, we can "unfold" the above recursive upper bound to the following, which is easy to verify by induction on k:

$$\eta_k \leq \sum_{j=1}^{k} \prod_{\ell=j+1}^{k} (1 + \varepsilon_\ell) 2\varepsilon_j \sin(3^{j-1}\theta). \tag{6}$$

For each $1 \leq j \leq k$, choose

$$\varepsilon_j = \frac{1}{100 \cdot 4^j}. \tag{7}$$

Note that $\sigma = \sum_{j=1}^{k} \varepsilon_k \leq 1/300$. With this choice of ε_j's, and the inequalities $1 + x \leq e^x$, $e^\sigma \leq 1.5$ and $\sin(x) \leq x$ for $x \leq \pi/2$ (which is the case here), we can upper bound the norm of the error term $|E_k\rangle$ after k iterations (see Equation (5)) as

$$\eta_k \leq \sum_{j=1}^{k} e^\sigma 2\varepsilon_j 3^{j-1}\theta \leq \frac{3\theta}{400} \sum_{j=1}^{k} (3/4)^{j-1} \leq \frac{3\theta}{100}. \tag{8}$$

Accordingly, up to very small error we have done perfect amplitude amplification.

A.3 Distributed amplitude amplification with imperfect reflection

We will now instantiate the above scheme to the case of *distributed* search, where our measure of cost is communication, that is, the number of qubits sent between Alice and Bob. Specifically, consider the *intersection problem* where Alice and Bob have inputs $x \in \{-1,1\}^n$ and $y \in \{-1,1\}^n$, respectively. Assume for simplicity that n is a power of 2, so $\log n$ is an integer. Alice and Bob want to find an $i \in \{0, \ldots, n-1\} = \{0,1\}^{\log n}$ such that $x_i = y_i = -1$, if such an i exists.

The basis states in this distributed problem are $|i\rangle|j\rangle$, and we define the set of "good" basis states as

$$\mathcal{G} = \{|i\rangle|j\rangle \mid x_i = y_j = -1\},$$

even though we are only looking for i, j where $i = j$ (it's easier to implement $O_\mathcal{G}$ with this more liberal definition of $\mathcal{G}$). Our protocol will start with the maximally entangled initial state $|\psi\rangle$ in n dimensions, which corresponds to $\log n$ EPR-pairs:

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i \in \{0,1\}^{\log n}} |i\rangle|i\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle,$$

where we assume there are t i's where $x_i = y_i = -1$, i.e., t solutions to the intersection problem, so

$$\theta = \arcsin(\sqrt{t/n}). \tag{9}$$

and

$$|G\rangle = \frac{1}{\sqrt{t}} \sum_{(i,i) \in \mathcal{G}} |i\rangle|i\rangle.$$

It costs $\lceil \log n \rceil$ qubits of communication between Alice and Bob to establish this initial shared state, or it costs nothing if we assume pre-shared entanglement. Our goal is to end up with a state that has large inner product with $|G\rangle$.

In order to be able to use amplitude amplification, we would like to be able to reflect about the above state $|\psi\rangle$. However, in general this perfect reflection R_ψ costs a lot of communication: Alice would send her $\log n$ qubits to Bob, who would unitarily put a -1 in front of all states orthogonal to $|\psi\rangle$, and then sends back Alice's qubits. This has a communication cost of $O(\log n)$ qubits, which is too much for our purposes. Fortunately, Theorem 19 gives us a way to implement a one-sided ε-error reflection protocol R_ψ^ε that only costs $O(\log(1/\varepsilon))$ qubits of communication.

The reflection $O_{\mathcal{G}}$ puts a "$-$" in front of the basis states $|i\rangle|j\rangle$ in $\mathcal{G}$. This can be implemented perfectly using only 2 qubits of communication, as follows. For the variables $x_i \in \{-1,1\}$, let $\widehat{x_i}$ denote their $\{0,1\}$-valued counterparts. That is, $\widehat{x_i} = 1$ if $x_i = -1$ and $\widehat{x_i} = 0$ if $x_i = 1$. To implement the reflection $O_{\mathcal{G}}$ on her basis state $|i\rangle$, Alice XORs $|\widehat{x_i}\rangle$ into a fresh auxiliary $|0\rangle$-qubit and sends this qubit to Bob. Bob receives this qubit and applies the following unitary map:

$$|b\rangle|j\rangle \mapsto y_j^b|b\rangle|j\rangle, \quad b \in \{0,1\}, j \in [n].$$

He sends back the auxiliary qubit. Alice sets the auxiliary qubit back to $|0\rangle$ by XOR-ing $\widehat{x_i}$ into it. Ignoring the auxiliary qubit (which starts and ends in state $|0\rangle$), this maps $|i\rangle|j\rangle \mapsto (-1)^{[x_i=y_j=-1]}|i\rangle|j\rangle$. Hence we have implemented $O_{\mathcal{G}}$ correctly: a minus sign is applied exactly for the good basis states, the ones where $x_i = y_j = -1$.

Now consider the algorithms (more precisely, communication protocols):

$$A_1 = R_\psi^{\varepsilon_1} \cdot O_{\mathcal{G}} \quad \text{and} \quad A_{j+1} = A_j R_\psi^{\varepsilon_{j+1}} A_j^* \cdot O_{\mathcal{G}} \cdot A_j$$

with the choice of ε_j's from Equation (7). If we pick $k = \lfloor \log_3(\pi/(2\theta)) \rfloor$, like in Equation (4), then $3^k\theta \in (\pi/6, \pi/2]$. Hence by Equation (5) and Equation (8), the inner product of our final state with $|G\rangle$ will be between $\sin(3^k\theta) - 3\theta/100 \geq 0.4$ and 1.

At this point Alice and Bob can measure, and with probability $\geq 0.4^2$ they will each see the same i, with the property that $x_i = y_i = -1$.

From Equation (3) and Theorem 19, the recursion for the communication costs of these algorithms is

$$C_{j+1} = 3C_j + O(\log(1/\varepsilon_{j+1})) + 2.$$

Solving this recurrence with our ε_j's from Equation (7) and the value of θ from Equation (9) we obtain

$$C_k = \sum_{j=1}^{k} 3^{k-j}(O(\log(1/\varepsilon_j)) + 2) = \sum_{j=1}^{k} 3^{k-j}O(j) = O(3^k) = O(\sqrt{n/t}).$$

Thus, using $O(\sqrt{n/t})$ qubits of communication we can find (with constant success probability) an intersection point i. This also allows us to solve the Set-Disjointness problem (the decision problem whose output is 1 if there is no intersection between x and y). Note that if the t we used equals the actual number of solutions only up to a factor of 2, the above protocol still has $\Omega(1)$ probability to find a solution, and $O(1)$ repetitions will boost this success probability to 0.99. In case we do not even know t approximately, we can use the standard technique of trying exponentially decreasing guesses for t to find an intersection point with communication $O(\sqrt{n})$.

Note that there is no log-factor in the communication complexity, in contrast to the original $O(\sqrt{n}\log n)$-qubit Grover-based quantum protocol for the intersection problem of Buhrman et al. [7]. Aaronson and Ambainis [1] earlier already managed to remove the log-factor, giving an $O(\sqrt{n})$-qubit protocol for Set-Disjointness as a consequence of their local version of quantum search on a grid graph (which is optimal [22]). We have just reproved this result of [1] in a different and arguably simpler way.

The above description is geared towards the intersection problem, where the "inner" function is $G = \mathsf{AND}_2$: we called a basis state $|i\rangle|j\rangle$ "good" if $x_i = y_j = -1$. However, this can easily be generalized to the situation where Alice and Bob's respective inputs are $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ and we want to find an $i \in [n]$ where $G(X_i, Y_i) = -1$

for some two-party function G, and define the set of “good” basis states as $\mathcal{G} = \{|i\rangle|j\rangle \mid G(X_i, Y_j) = -1\}$.[4] The only thing that changes in the above is the implementation of the reflection $O_{\mathcal{G}}$, which would now be computed by means of an exact quantum communication protocol for $G(X_i, Y_j)$, at a cost of $2\mathsf{Q}_E^{cc}(G)$ qubits of communication.[5] Note that because we can check (at the expense of another $\mathsf{Q}_E^{cc}(G)$ qubits of communication) whether the output index i actually satisfies $G(X_i, Y_i) = -1$, we may assume the protocol has one-sided error: it always outputs “no” if there is no such i. This concludes the proof of Theorem 21.

[4] We intentionally use the letter “G” to mean “good” in $\mathcal{G}$ and and to refer to the two-party function G, since G determines which basis states $|i\rangle|j\rangle$ are “good.”

[5] The factor of 2 is to reverse the protocol after the phase $G(X_i, Y_j)$ has been added to basis state $|i\rangle|j\rangle$, in order to set any workspace qubits back to $|0\rangle$.

Near-Optimal Algorithms for Point-Line Covering Problems

Jianer Chen
Department of Computer Science and Engineering, Texas A&M University, TX, USA

Qin Huang
Department of Computer Science and Engineering, Texas A&M University, TX, USA

Iyad Kanj
School of Computing, DePaul University, Chicago, IL, USA

Ge Xia
Department of Computer Science, Lafayette College, Easton, PA, USA

Abstract

We study fundamental point-line covering problems in computational geometry, in which the input is a set S of points in the plane. The first is the Rich Lines problem, which asks for the set of all lines that each covers at least λ points from S, for a given integer parameter $\lambda \geq 2$; this problem subsumes the 3-Points-on-Line problem and the Exact Fitting problem, which – the latter – asks for a line containing the maximum number of points. The second is the NP-hard problem Line Cover, which asks for a set of k lines that cover the points of S, for a given parameter $k \in \mathbb{N}$. Both problems have been extensively studied. In particular, the Rich Lines problem is a fundamental problem whose solution serves as a building block for several algorithms in computational geometry.

For Rich Lines and Exact Fitting, we present a randomized Monte Carlo algorithm that achieves a lower running time than that of Guibas et al.'s algorithm [*Computational Geometry* 1996], for a wide range of the parameter λ. We derive lower-bound results showing that, for $\lambda = \Omega(\sqrt{n \log n})$, the upper bound on the running time of this randomized algorithm matches the lower bound that we derive on the time complexity of Rich Lines in the algebraic computation trees model.

For Line Cover, we present two kernelization algorithms: a randomized Monte Carlo algorithm and a deterministic algorithm. Both algorithms improve the running time of existing kernelization algorithms for Line Cover. We derive lower-bound results showing that the running time of the randomized algorithm we present comes close to the lower bound we derive on the time complexity of kernelization algorithms for Line Cover in the algebraic computation trees model.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases line cover, rich lines, exact fitting, kernelization, randomized algorithms, complexity lower bounds, algebraic computation trees

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.21

Related Version *Full Version*: https://arxiv.org/abs/2012.02363

1 Introduction

We study fundamental problems in computational geometry pertaining to covering a set S of n points in the plane with lines. The first problem, referred to as Rich Lines, is defined as:

> Rich Lines: Given a set S of n points and an integer parameter $\lambda \geq 2$, compute the set of lines that each covers at least λ points.

A special case of Rich Lines that has received attention is the Exact Fitting problem [25], which asks for computing a line that covers the maximum number of points in S. Exact Fitting subsumes the well-known 3-Points-on-Line problem in an obvious way.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 21; pp. 21:1–21:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The Rich Lines problem is a fundamental problem whose solution serves as a building block for several algorithms in computational geometry [17, 16, 21, 23, 24, 31], including algorithms for the fundamental Line Cover problem, which is our other focal problem:

> Line Cover: Given a set S of n points and a parameter $k \in \mathbb{N}$, decide if there exist at most k lines that cover all points in S.

See Figure 1 for an illustration of Rich Lines, Exact Fitting, and Line Cover.

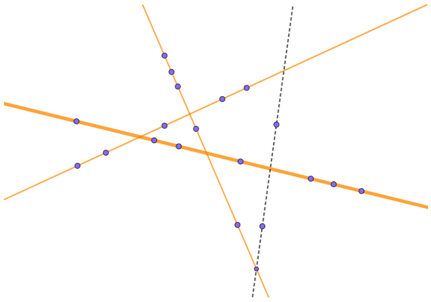


Figure 1 Illustration of an instance of Line Cover with $k = 4$, an instance of Rich Lines with $\lambda = 5$, and an instance of Exact Fitting. The set of all lines is the solution to the Line Cover instance; the set of solid orange lines is the solution to the Rich Lines instance; and the bold orange line is a solution to the Exact Fitting instance.

The Line Cover problem is NP-hard [33], and has been extensively studied in parameterized complexity [1, 7, 16, 23, 29, 31, 41], especially with respect to kernelization. Guibas et al.'s algorithm [25] for Rich Lines was used to give a simple kernelization algorithm that computes a kernel of at most k^2 points, and this upper bound on the kernel size was proved to be essentially tight by Kratsch et al. [29].

The current paper derives both upper and lower bounds on the time complexity of Rich Lines, and the time complexity of the kernelization of Line Cover. Most of the algorithmic upper-bound results we present are randomized Monte Carlo algorithms, providing guarantees on the running time of the algorithms, but may make one-sided errors with a small probability. Our work is motivated by the applications of both problems to on-line data analytics [35], where massive data processing within a guaranteed time upper bound is required (e.g., dynamic or streaming environments [3, 8, 35]). In such settings, where the data set has an enormous size, classical algorithmic techniques become infeasible, and timely pre-processing the very large input in order to reduce its size becomes essential. Therefore, we seek algorithms whose running time is nearly linear and whose space complexity is low, trading off the optimality/correctness of the algorithm with a small probability.

1.1 Related Work

Both Rich Lines and Exact Fitting were studied by Guibas et al. [25], motivated by their applications in statistical analysis (e.g., linear regressions), computer vision, pattern recognition, and computer graphics [26, 27]. Guibas et al. [25] developed an $\mathcal{O}(\min\{\frac{n^2}{\lambda}\log\frac{n}{\lambda}, n^2\})$-time deterministic algorithm for Rich Lines, and used it to solve Exact Fitting within the same time upper bound. Guibas et al.'s algorithm [25] was subsequently used in many algorithmic results [17, 16, 21, 23, 24, 31] pertaining to geometric covering problems and their applications.

The Line Cover problem has been extensively studied with respect to several computational frameworks, including approximation [7, 23] and parameterized complexity [1, 7, 16, 29, 31, 41]. The problem is known to be APX-hard [30] and is approximable within ratio $\log n$, being a special case of the set cover problem [28].

From the parameterized complexity perspective, several fixed-parameter tractable algorithms for Line Cover were developed [1, 23, 31, 41], leading to the current-best algorithm that runs in time $(c \cdot k/\log k)^k n^{\mathcal{O}(1)}$ [1], for some constant $c > 0$. Guibas et al.'s algorithm [25] was used in several works to give a kernel of size k^2 that is computable in time $\mathcal{O}(\min\{\frac{n^2}{k}\log(\frac{n}{k}), n^2\})$ [16, 29, 31]. This quadratic kernel size was shown to be essentially tight by Kratsch et al. [29], who showed that: For any $\epsilon > 0$, unless the polynomial-time hierarchy collapses to the third level, Line Cover has no kernel of size $\mathcal{O}(k^{2-\epsilon})$.

Kernelization algorithms for Line Cover have drawn attention in recent research in massive-data processing; Mnich [35] discusses how the Line Cover problem is used in such settings, where the point set represents a very large collection of observed (accurate) data, and the solution sought is a model consisting of at most k linear predictors [20].

Chitnis et al. [8] studied Line Cover in the streaming model, and Alman et al. [3] studied the problem in the dynamic model. We mention that Chitnis et al.'s streaming algorithm [8] may be used to give a Monte Carlo kernelization algorithm for Line Cover running in time $\mathcal{O}(n(\log n)^{\mathcal{O}(1)})$, and the dynamic algorithm of Alman et al. [3] may be used to give a deterministic kernelization algorithm for Line Cover running in time $\mathcal{O}(nk^2)$.

We finally note that there has been considerable work on randomized algorithms for geometric problems (see [2, 9, 10, 12], to name a few). The most relevant of which to our work is the randomized algorithm for approximating geometric set covering problems [5, 11] (see also [2]), which implies an $\mathcal{O}(\log k)$-factor approximation algorithm for the optimization version of Line Cover whose expected running time is $\mathcal{O}(nk(\log n)(\log k))$.

1.2 Results and Techniques

In this paper, we develop new tools to derive upper and lower bounds on the time complexity of Rich Lines and the kernelization time complexity of Line Cover. Our results and techniques are summarized as follows.

1.2.1 Results for Rich Lines

We present a randomized one-sided errors Monte Carlo algorithm for Rich Lines that, with probability at least $1 - \frac{3}{n^2}$, returns the correct solution set, where n is the number of points. The algorithm achieves a lower running time upper bound than Guibas et al.'s algorithm [25] for a wide range of the parameter λ, namely for $\lambda = \Omega(\log n)$, and matches its running time otherwise. For instance, when $\lambda = \Theta(\sqrt{n \log n})$, the running time of our algorithm is $\mathcal{O}(n \log n)$, whereas that of Guibas et al.'s algorithm is $\mathcal{O}(n^{3/2}\sqrt{\log n})$, yielding a $(\sqrt{n/\log n})$-factor improvement. We show that, for $\lambda = \Omega(\sqrt{n \log n})$, the upper bound of $\mathcal{O}(n \log(\frac{n}{\lambda}))$ on the running time of our randomized algorithm matches the lower bound that we derive on the time complexity of the problem in the algebraic computation trees model. The algorithm for Rich Lines implies an algorithm for Exact Fitting with the same performance guarantees – as shown by Guibas et al. [25], obtained by binary-searching for the value of λ that corresponds to the line(s) containing the maximum number of points.

The crux of the technical contributions leading to the randomized algorithm we present is a set of new tools we develop pertaining to point-line incidences and sampling. The aforementioned tools allow us to show that, by sampling a smaller subset of the original set of points, with high probability, we can reduce the problem of computing the set of λ-rich lines in the original set to that of computing the set of λ'-rich lines in the smaller subset, where λ' is a smaller parameter than λ.

The time lower-bound result we present is obtained via a 2-step reduction. The first employs Ben-Or's framework [4] to show a time lower bound of $\Omega(n\log(\frac{n}{\lambda}))$ in the algebraic computation trees model on a problem that we define, referred to as the Multiset Subset Distinctness problem. We then compose this reduction with a reduction from Multiset Subset Distinctness to Rich Lines, thus establishing the $\Omega(n\log(\frac{n}{\lambda}))$ lower-bound result for Rich Lines. We note that these reductions are very "sensitive", and hence need to be crafted carefully, as the lower-bound results apply for *every* value of n and λ.

1.2.2 Results for Line Cover

We derive a lower bound on the time complexity of kernelization algorithms for Line Cover in the algebraic computation trees model and show that the running time of any such algorithm must be at least $cn\log k$ for some constant $c > 0$; more specifically, one cannot asymptotically improve either of the two factors n or $\log k$ in this term. This result particularly rules out the possibility of a kernelization algorithm that runs in $\mathcal{O}(n)$ time (i.e., in linear time). We derive this lower bound by combining a lower-bound result by Grantson and Levcopoulos [23] on the time complexity of Line Cover with a result that we prove in this paper connecting the time complexity of Line Cover to its kernelization time complexity. In fact, it is not difficult to develop a kernelization algorithm for Line Cover that runs in time $\mathcal{O}(n\log k + g(k))$ for some computable function $g(k)$, and computes a kernel of size k^2. This can be done by processing the input in "batches" of size roughly k^2 each; this is implied by the algorithm in [23], which runs in time $\Omega(n\log k + k^4\log k)$, and approximates the optimization version of Line Cover. Therefore, we focus on developing kernelization algorithms where the function $g(k)$ in their running time is as small as possible. Since we can assume that $n \geq k^2$ (otherwise, the instance is already kernelized), we may assume that $g(k) = \Omega(k^2\log k)$. Therefore, we endeavor to develop a kernelization algorithm for which the function $g(k)$ – in its running time – is as close as possible to $\mathcal{O}(k^2\log k)$, and hence, a kernelization algorithm whose running time is as close as possible to $\mathcal{O}(n\log k)$. In addition, reducing the function $g(k)$ serves well our purpose of obtaining near-linear-time kernelization algorithms for Line Cover due to their potential applications [20, 35].

We present two kernelization algorithms for Line Cover. The first is a randomized one-sided errors Monte Carlo algorithm that runs in time $\mathcal{O}(n\log k + k^2(\log^2 k)(\log\log k)^2)$ and space $\mathcal{O}(k^2\log^2 k)$ and, with probability at least $1 - \frac{2}{k^3}$, computes a kernel of size at most k^2. The second is a deterministic algorithm that computes a kernel of size at most k^2 in time $\mathcal{O}(n\log k + k^3(\log^3 k)(\sqrt{\log\log k}))$. Both algorithms improve the running time of existing kernelization algorithms for Line Cover [3, 8, 16, 23, 29, 31]. Moreover, the running time of the randomized algorithm comes very close to the derived lower bound on the time complexity of kernelization algorithms for Line Cover, as it only differs from it by a factor of $\log k(\log\log k)^2$ in the term that depends only on k.

The key tool leading to the improved kernelization algorithms is partitioning the "saturation range" of the saturated lines (i.e., the lines that each contains at least $k + 1$ points and must be in the solution) in the batch of points under consideration into intervals, thus defining a spectrum of saturation levels. Then the algorithm for Rich Lines (either the randomized or Guibas et al.'s algorithm [25]) is invoked starting with the highest saturation threshold, and iteratively decreasing the threshold until either: the saturated lines computed cover "enough" points of the batch under consideration, or the total number of saturated lines computed is "large enough" thus making "enough progress" towards computing the line cover. This scheme enables us to amortize the running time of the algorithm that computes the saturated lines, creating a win/win situation and improving the overall running time.

2 Preliminaries

We assume familiarity with basic geometry, probability, and parameterized complexity and refer to the following standard textbooks on some of these subjects [13, 14, 18, 34, 36]. For a positive integer i, we write $[i]$ for $\{1, 2, \ldots, i\}$. We write "*w.h.p.*" as an abbreviation for "with high probability", and we write "*u.a.r.*" as an abbreviation for "uniformly at random".

Probability. The *union bound* states that, for any probabilistic events $E_1, E_2, \ldots, E_j$, we have: $\Pr\left(\bigcup_{i=1}^{j} E_i\right) \leq \sum_{i=1}^{j} \Pr(E_i)$. For any discrete random variables $X_1, \ldots, X_n$ with finite expectations, it is well known that: $E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i]$, where $E[X]$ denotes the expectation of X.

The following lemma, for the sum of a random sample without replacement from a finite set, can be viewed as an application of Chernoff's bounds – customized to our needs – to negatively correlated random variables:

▶ **Lemma 1.** *Let $\mathcal{C} = \{x_1, \ldots, x_N\}$, where $x_i \in \{0, 1\}$ for $i \in [N]$. Let $X_1, X_2, \ldots, X_j$ denote a random sample without replacement from $\mathcal{C}$. Let $X = \sum_{i=1}^{j} X_i$, $\mu = E[X]$, and μ_1, μ_2 be any two values such that $\mu_1 \leq \mu \leq \mu_2$. Then, (A) for any $\delta > 0$, we have $\Pr(X \geq (1+\delta)\mu_2) \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu_2}$; and (B) for any $0 < \delta < 1$, we have $\Pr(X \leq (1-\delta)\mu_1) \leq \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu_1}$.*

Point-Line Incidences. Let S be a set of points. A line l *covers* a point $p \in S$ if l passes through p (i.e., contains p). A set L of lines *covers* S if every point in S is covered by at least one line in L. A line l is *induced* by S if l covers at least 2 points of S, and a set L of lines is *induced* by S if every line in L is induced by S. For a set L of lines, we define $I(L, S)$ as $I(L, S) = |\{(q, l) \mid q \in S \cap l, l \in L\}|$; that is, $I(L, S)$ is the number of incidences between L and S. For a line l, let $I(l, S) = |\{(q, l) \mid q \in S \cap l\}|$.

The following theorems upper bound $I(L, S)$ and the complexity of computing it:

▶ **Theorem 2** ([37])**.** *$I(L, S) \leq \frac{5}{2}(mn)^{2/3} + m + n$, where $n = |S|$ and $m = |L|$.*

The theorem below follows from Theorem 3.1 in [32] after a slight modification, as was also observed by [15]:

▶ **Theorem 3** ([15, 32])**.** *Let S be a set of n points and L a set of m lines in the plane. The set of incidences between S and L, and hence $I(L, S)$, can be computed in (deterministic) time $\mathcal{O}(n \log m + m \log n + (mn)^{2/3} 2^{\mathcal{O}(\log^*(n+m))})$. Moreover, within the same running time, we can compute for each line $l \in L$ the set of points in S that are contained in l.*

Let P be a subset of S, and let $x \in \mathbb{N}$. We say that a line l is *x-rich* for P if l covers at least x points from P; when P is clear from the context, we will simply say that l is x-rich.

The following extends Theorem 2 in [40], which applies only when the constant $c < 1$:

▶ **Theorem 4.** *Let S be a set of n points, let $c > 0$ be a constant, and let k be an integer such that $2 \leq k \leq c\sqrt{n}$. Let L be the set of k-rich lines for S. Then $|L| < \max\{\frac{40n^2}{k^3}, \frac{40c^2n^2}{k^3}\}$.*

Parameterized Complexity. A *parameterized problem* Q is a subset of $\Omega^* \times \mathbb{N}$, where Ω is a fixed alphabet. Each instance of Q is a pair (I, κ), where $\kappa \in \mathbb{N}$ is called the *parameter*. A parameterized problem Q is *fixed-parameter tractable* (FPT), if there is an algorithm, called an FPT-*algorithm*, that decides whether an input (I, κ) is a member of Q in time $f(\kappa) \cdot |I|^{\mathcal{O}(1)}$, where f is a computable function and $|I|$ is the input instance size. The class FPT denotes the class of all fixed-parameter tractable parameterized problems. A parameterized problem is *kernelizable* if there exists a polynomial-time reduction that maps an instance (I, κ) of the problem to another instance (I', κ') such that (1) $|I'| \le f(\kappa)$ and $\kappa' \le f(\kappa)$, where f is a computable function, and (2) (I, κ) is a yes-instance of the problem if and only if (I', κ') is. The instance (I', κ') is called the *kernel* of I.

3 A Randomized Algorithm for Rich Lines

In this section, we present a randomized algorithm for Rich Lines that achieves a better running time than Guibas et al.'s algorithm [25] for a wide range of the parameter λ (in the problem definition). We will show in Section 5 that for $\lambda = \Omega(\sqrt{n \log n})$, the upper bound of $\mathcal{O}(n \log(\frac{n}{\lambda}))$ on the running time of our randomized algorithm for Rich Lines matches the lower bound on its time complexity that we derive in the algebraic computation trees model.

We first present an intuitive low-rigor explanation of the randomized algorithm and the techniques entailed. The crux of the technical results in this section lies in Lemma 6. This lemma shows that, by sampling a smaller subset S' of S whose size depends on λ, w.h.p. we can reduce the problem of computing the set of λ-rich lines for S to that of computing the set of λ'-rich lines for S', where $\lambda' < \lambda$.

The algorithm exploits the above technical results as follows. Given an instance (S, λ) of Rich Lines, the algorithm samples a subset $S' \subseteq S$ whose size depends on λ. Depending on the value of λ, the algorithm defines a threshold value λ', and computes the set L' of λ'-rich lines for S'. As we show in this section, w.h.p. L' contains all the λ-rich lines for S, and hence, by sifting through the lines in L', the algorithm computes the solution to (S, λ).

▶ **Lemma 5.** *Let $\lambda \ge 2\sqrt{n}$. The number of λ-rich lines for S is at most $\frac{2n}{\lambda}$.*

Proof. Let $L = \{l_1, l_2, ..., l_m\}$ be the set of λ-rich lines for P. Denote by z_i, for $i = 1, 2, \dots, m$, the number of points covered by l_i. The set L covers at least $\sum_{i=1}^{m} z'_i$ points, where $z'_i = \max\{z_i - i + 1, 0\}$. This is true since l_1 covers at least z_1 points, l_2 covers at least $z_2 - 1$ new points (excluding at most 1 point on l_1), and in general, l_i covers at least $z_i - i + 1$ new points (excluding at most $i - 1$ points covered by $\{l_1, ..., l_{i-1}\}$) if $i \le z_i$ or 0 otherwise. Suppose, to get a contradiction, that $m > \frac{2n}{\lambda}$, and consider the first $m' = \lceil \frac{2n}{\lambda} \rceil$ lines. Then $\sum_{i=1}^{m} z'_i \ge \sum_{i=1}^{m'} z'_i = \sum_{i=1}^{m'} (z_i - i + 1)$ since $i \le m' < z_i$. Hence, $\sum_{i=1}^{m'} z'_i \ge \lambda \cdot m' - \frac{(m'-1)m'}{2}$ $\ge 2n - \frac{(\frac{2n}{\lambda}+1)\frac{2n}{\lambda}}{2} = 2n - \frac{n}{\lambda} - \frac{2n^2}{\lambda^2}$. Since $\lambda \ge 2\sqrt{n}$, $\sum_{i=1}^{m'} z'_i \ge 2n - \frac{\sqrt{n}}{2} - \frac{n}{2} > n$ (assuming w.l.o.g. that $n > 1$), which is a contradiction. Therefore, we have $m \le \frac{2n}{\lambda}$. ◀

Throughout this section, S denotes a set of $n \ge 3$ points. Let $S'(m)$ be a set formed by sampling without replacement $m \le n$ points from S uniformly and independently at random.

▶ **Lemma 6.** *Let λ be an integer satisfying $140 \ln^{3/2} n \le \lambda \le n$. Let $S'(m)$ be as defined above where $m = \lceil \frac{140 n \ln n}{\lambda} \rceil$. Let L_1 be the set of λ-rich lines for S, and let L_3 be the set of $(98 \ln n)$-rich lines for $S'(m)$. Then, with probability at least $1 - \frac{2}{n^2}$, we have: (1) $L_1 \subseteq L_3$, and (2) if $\lambda \ge 5\sqrt{n}$, then $|L_3| \le \frac{5n}{\lambda}$; if $\lambda < 5\sqrt{n}$, then $|L_3| < \frac{2500 n^2}{\lambda^3}$.*

Proof. Let L_2 be the set of lines induced by S containing less than $\frac{2\lambda}{5}$ points each. Without loss of generality, suppose that $L_1 = \{l_1, \ldots, l_d\}$ and $L_2 = \{l_{d+1}, \ldots, l_{d'}\}$, and note that $d \le d' \le \binom{n}{2}$. For $i \in [d']$, let x_i be the number of points in S covered by l_i, and let X'_i be the random variable, where X'_i is the number of the points in $S'(m)$ on l_i.

For $i \in [d]$, we have $E[X'_i] = \frac{x_i}{n} \cdot m \ge 140 \ln n$ since $x_i \ge \lambda$. Applying part (B) of the Chernoff bounds in Lemma 1 with $\mu_1 = 140 \ln n$, we have $\Pr(X'_i \le (1 - 3/10) \cdot 140 \ln n) \le \left(\frac{e^{-3/10}}{(1-3/10)^{1-3/10}}\right)^{140 \ln n} \le \frac{1}{n^4}$, where the last inequality can be easily verified by a simple analysis. Let E_i, for $i \in [d]$, denote the event that $X'_i \le (1 - 3/10) \cdot 140 \ln n$. Applying the union bound, we have $\Pr(\bigcap_{i=1}^{d} \overline{E_i}) = 1 - \Pr(\bigcup_{i=1}^{d} E_i) \ge 1 - d \cdot \frac{1}{n^4} \ge 1 - \frac{1}{n^2}$. Let $\mathcal{E} = \bigcap_{i=1}^{d} \overline{E_i}$. The probability that every line $l_i \in L_1$ contains at least $98 \ln n$ points of $S'(m)$ is at least $1 - \frac{1}{n^2}$. That is to say, with probability at least $1 - \frac{1}{n^2}$, we have $L_1 \subseteq L_3$.

For $i = d+1, \ldots, d'$, $E[X'_i] = \frac{x_i}{n} \cdot m < \frac{x_i}{n}(\frac{140 n \ln n}{\lambda} + 1) \le 56 \ln n + \frac{2\lambda}{5n} < 57 \ln n$, since $x_i \le \frac{2\lambda}{5}$ and $140 \ln^{3/2} n \le \lambda \le n$. Applying part (A) of the Chernoff bounds in Lemma 1 with $\mu_2 = 57 \ln n$, we get $\Pr(X'_i \ge (1 + \frac{13}{19}) \cdot 57 \ln n) \le \left(\frac{e^{13/19}}{(1+13/19)^{1+13/19}}\right)^{57 \ln n} \le \frac{1}{n^4}$, where the last inequality can be easily verified by a simple analysis. Consequently, via the union bound, the probability that every line $l_i \in L_2$ contains less than $96 \ln n$ sampled points is at least $1 - (d' - d) \cdot \frac{1}{n^4} \ge 1 - \frac{1}{n^2}$. It follows that, with probability at least $1 - \frac{1}{n^2}$, we have $L_3 \cap L_2 = \emptyset$.

Altogether, with probability at least $1 - \frac{2}{n^2}$, $L_1 \subseteq L_3$ and $L_2 \cap L_3 = \emptyset$. Recall that each line in L_3 covers at least $\frac{2\lambda}{5}$ points of S. If $\frac{2\lambda}{5} \ge 2\sqrt{n}$, i.e., $\lambda \ge 5\sqrt{n}$, we have $|L_3| \le \frac{2n}{2\lambda/5} = \frac{5n}{\lambda}$ by Lemma 5. If $\frac{2\lambda}{5} < 2\sqrt{n}$, i.e. $\lambda < 5\sqrt{n}$, by Theorem 4 (with $c \le 2$), we have $|L_3| < \frac{40 \cdot 2^2 n^2}{(2\lambda/5)^3} = \frac{2500 n^2}{\lambda^3}$. It follows that, with probability at least $1 - \frac{2}{n^2}$, parts (1) and (2) of the lemma hold. ◀

■ **Algorithm 1 : Alg-RichLines**(S, λ) – A randomized algorithm for computing all λ-rich lines.

```
Input: a set of points $S$ and $\lambda \in \mathbb{N}$.
Output: The set $L$ of $\lambda$-rich lines for $S$.
1: if $\lambda < \ln n$ then apply Guibas et al.'s algorithm [25] to compute $L$ and return $L$;
2: sample $x = \lceil \frac{10 n^2 \ln n}{\lambda^2} \rceil$ pairs of points $(p_1, q_1), \ldots, (p_x, q_x)$ u.a.r. from $\binom{S}{2}$;
3: let $l_i$ be the line formed by $(p_i, q_i)$, for $i \in [x]$; let $Q_1$ be the multi-set $\{l_1, l_2, \ldots, l_x\}$, and
   let $Q_2$ be the set of distinct lines in $Q_1$;
4: if $\lambda \le 140 \ln^{3/2} n$ then let $L = \{l \in Q_2 \mid I(l, S) \ge \lambda\}$; return $L$;
5: let $m = \lceil \frac{140 n \ln n}{\lambda} \rceil$, $y = 98 \ln n$;
6: sample $m$ points u.a.r. from $S$ without replacement to obtain $S'(m)$;
7: if $\lambda < 5\sqrt{n}$ then
8:     let $z = \frac{2500 n^2}{\lambda^3}$;
9: else let $z = \frac{5n}{\lambda}$;
10: let $L' = \{l \in Q_2 \mid I(l, S'(m)) \ge y\}$;
11: if $|L'| \le z$ then let $L = \{l \in L' | I(l, S) \ge \lambda\}$; return $L$;
12: else return $\emptyset$;
```

Refer to **Alg-RichLines** for the terminologies used in the subsequent discussions.

▶ **Lemma 7.** *Let $\lambda \in \mathbb{N}$. Let L_1 be the set of λ-rich for S. Then, with probability at least $1 - \frac{3}{n^2}$,* **Alg-RichLines**(S, λ) *returns a set $L = L_1$.*

Proof. If $\lambda < \ln n$ then $L = L_1$ with probability 1 by Step 1, as Guibas et al.'s algorithm [25] computes L_1 deterministically.

Now consider the case that $\lambda \geq \ln n$. Let l be an arbitrary line in L_1. Step 2 samples x pairs of points that determine x lines. In a single sampling, the probability ρ that l is sampled is $\binom{\lambda}{2}/\binom{n}{2} \geq \frac{\lambda^2}{2n^2}$ because l covers at least λ points of S. Thus, $\Pr(l \notin Q_1) = (1-\rho)^x \leq e^{-\rho x} \leq \frac{1}{n^5}$. Since $|L_1| < n^2$, applying the union bound, we get $Pr(L_1 \subseteq Q_1) \geq 1 - \frac{1}{n^3}$. Hence, we have $\Pr(L_1 \subseteq Q_2) \geq 1 - \frac{1}{n^3}$ since Q_2 is obtained from Q_1 by removing repeated lines. Let L_3 be the set of y-rich lines for $S'(m)$. If $\lambda < 140 \ln^{3/2} n$, then since $L_1 \subseteq Q_2$ with probability at least $1 - \frac{1}{n^3}$, the algorithm returns in Step 4 a set L that, with probability at least $1 - \frac{1}{n^3}$, is equal to L_1 .

Finally, if $140 \ln^{3/2} n \leq \lambda$, then by Lemma 6, $L_1 \subseteq L_3$ and $|L_3| \leq z$ with probability at least $1 - \frac{2}{n^2}$. Since $L' = L_3 \cap Q_2$, $|L'| \leq z$ holds with probability at least $1 - \frac{2}{n^2}$. Since $\Pr(L_1 \subseteq Q_2) \geq 1 - \frac{1}{n^3}$, we have $L_1 \subseteq (L_3 \cap Q_2)$ and $|L'| \leq z$ with probability at least $1 - \frac{3}{n^2}$. Thus, $\Pr(L_1 \subseteq L') \geq 1 - \frac{3}{n^2}$. Therefore, the algorithm returns in Step 11 a set L equal to L_1 with probability at least $1 - \frac{3}{n^2}$. ◀

The correctness of **Alg-RichLines**, stated in the following theorem, follows from Lemma 7.

▶ **Theorem 8.** *Let S be a set of n points and $\lambda \in \mathbb{N}$. With probability at least $1 - \frac{3}{n^2}$,* **Alg-RichLines**(S, λ) *solves the* Rich Lines *problem in time:*

(1) $\mathcal{O}(n^2)$ *if $\lambda < \ln n$; and*

(2) $\mathcal{O}(n \log \frac{n}{\lambda} + \frac{n^2 \log n \log \frac{n}{\lambda}}{\lambda^2})$ *otherwise.*

Proof. By Lemma 7, with probability at least $1 - \frac{3}{n^2}$, **Alg-RichLines**(P, λ) correctly returns the set of λ-rich lines for P. We discuss next the running time of the algorithm.

Case 1: $\lambda < \ln n$. In this case the running time of the algorithm is that of Guibas et al.'s algorithm [25], which is $\mathcal{O}(n^2)$.

It is easy to see that Step 2 takes $\mathcal{O}(x) = \mathcal{O}(\frac{n^2 \ln n}{\lambda^2})$ time. Step 3 can be implemented by sorting the slopes of the x lines, which takes $\mathcal{O}(x \ln x) = O(\frac{n^2 \ln n}{\lambda^2} \cdot (\ln \frac{n}{\lambda} + \ln \ln n))$ time. Step 6 takes time $\mathcal{O}(m) = \mathcal{O}(\frac{140 n \ln n}{\lambda}) = \mathcal{O}(n)$ since $n \geq \lambda \geq 140 \ln^{3/2} n$. Steps 5, 7, 8, 9 and 12 take constant time. Note that all the above running times (for Steps 2, 3, 5, 6, 7, 8, 9, 12) are dominated by the running time listed in item (2) of the theorem.

We discuss the running time of Step 4 in Case 2 below, and that of Step 10 and Step 11 in both Cases 3 and 4. Note that, to determine the set of rich lines for S in Steps 4, 10 and 11, we apply Theorem 3 to compute the number of points in S (or $S'(m)$) on each of the lines in question, thus determining the set of rich lines for S (or $S'(m)$).

Case 2: $\ln n \leq \lambda \leq 140 \ln^{3/2} n$. In this case, Step 4 takes time $T_1 = \mathcal{O}(x \log n + n \log x + (nx)^{2/3} 2^{\mathcal{O}(\log^*(x+n))})$ by Theorem 3. Substituting $x = \lceil \frac{10n^2 \ln n}{\lambda^2} \rceil$, we obtain $T_1 = \mathcal{O}(\frac{n^2 \ln^2 n}{\lambda^2} + n \ln n + \frac{n^2 \ln^{2/3} n}{\lambda^{4/3}} 2^{\mathcal{O}(\log^* n)}) = \mathcal{O}(\frac{n^2 \ln^2 n}{\lambda^2})$, which is dominated by the running time listed in item (2) of the theorem.

We discuss the running time of Step 10 in both Case 3 and 4. Note that $|S'(m)| = m = \lceil \frac{140 n \ln n}{\lambda} \rceil$ and $|Q_2| \leq x = \mathcal{O}(\frac{n^2 \ln n}{\lambda^2})$. By Theorem 3, Step 10 takes time:

$$\begin{aligned} T_2 &= \mathcal{O}(m \log x + x \log m + (xm)^{2/3} 2^{\mathcal{O}(\log^*(x+m))}) \\ &= \mathcal{O}(\frac{n^2 \ln n}{\lambda^2}(\ln(\frac{n}{\lambda}) + \ln \ln n) + \frac{n^2}{\lambda^2} \ln^{4/3}(n) 2^{\mathcal{O}(\log^* n)}). \end{aligned}$$

If $\lambda \leq n^{2/3}$, we have $\frac{n^2}{\lambda^2} \ln^{4/3}(n) 2^{\mathcal{O}(\log^* n)} = \mathcal{O}(\frac{n^2 \log n \log \frac{n}{\lambda}}{\lambda^2})$, and if $n^{2/3} < \lambda \leq n$, we have $\frac{n^2}{\lambda^2} \ln^{4/3}(n) 2^{\mathcal{O}(\log^* n)} = \mathcal{O}(n \log \frac{n}{\lambda})$. Altogether, $\frac{n^2}{\lambda^2} \ln^{4/3}(n) 2^{\mathcal{O}(\log^* n)} = \mathcal{O}(n \log \frac{n}{\lambda} + \frac{n^2 \log n \log \frac{n}{\lambda}}{\lambda^2})$. Therefore, $T_2 = \mathcal{O}(n \log \frac{n}{\lambda} + \frac{n^2 \log n \log \frac{n}{\lambda}}{\lambda^2})$, which is dominated by the running time listed in item (2) of the theorem.

Case 3: $140\ln^{3/2} n < \lambda < 5\sqrt{n}$. Step 11 takes time $T_3 = \mathcal{O}(n\log|L'| + |L'|\log n + (n|L'|)^{2/3}2^{\mathcal{O}(\log^*(n+|L'|))})$ by Theorem 3. Since $|L'| \leq z = \frac{2500n^2}{\lambda^3}$, we have $T_3 = \mathcal{O}(n\ln n + \frac{n^2\log n}{\lambda^3} + \frac{n^2}{\lambda^2}2^{\mathcal{O}(\log^* n)}) = \mathcal{O}(n\log\frac{n}{\lambda} + \frac{n^2\log n\log\frac{n}{\lambda}}{\lambda^2})$. Thus, the total running time in this case is $\mathcal{O}(n\log\frac{n}{\lambda} + \frac{n^2\log n\log\frac{n}{\lambda}}{\lambda^2})$.

Case 4: $\lambda \geq 5\sqrt{n}$. Step 11 takes time $T_4 = \mathcal{O}(n\log|L'| + |L'|\log n + (n|L'|)^{2/3}\cdot 2^{\mathcal{O}(\log^*(n+|L'|))})$ by Theorem 3. Since $|L'| \leq z = \mathcal{O}(\frac{n}{\lambda})$, $T_4 = \mathcal{O}(n\log\frac{n}{\lambda} + \frac{n}{\lambda}\log n + \frac{n^{4/3}}{\lambda^{2/3}}2^{\mathcal{O}(\log^* n)}) = \mathcal{O}(n\log\frac{n}{\lambda} + \frac{n^{4/3}}{\lambda^{2/3}}2^{\mathcal{O}(\log^* n)}) = \mathcal{O}(n\log\frac{n}{\lambda})$. The last equality holds because $\lambda \geq 5\sqrt{n}$. Consequently, the total running time in this case is $\mathcal{O}(n\log\frac{n}{\lambda} + \frac{n^2\log n\log\frac{n}{\lambda}}{\lambda^2})$. ◀

Guibas et al.'s algorithm [25] solves the Rich Lines and the Exact Fitting problems in the plane in time $\mathcal{O}(\min\{\frac{n^2}{\lambda}\log\frac{n}{\lambda}, n^2\})$. Theorem 8 is an improvement over Guibas et al.'s algorithm [25] for both problems for all values of $\lambda \geq \ln n$, and for $\lambda < \ln n$ it obviously has a matching running time. In particular, for $\ln n \leq \lambda \leq 140\ln^{3/2} n$, the improvement could be in the order of $\frac{1}{\sqrt{\log n}}$ (i.e., the running time of **Alg-RichLines** is a $\frac{1}{\sqrt{\log n}}$-fraction of that in [25]); for $140\ln^{3/2} n < \lambda < 5\sqrt{n}$, the improvement could be in the order of $\frac{\log n}{\sqrt{n}}$; and for $\lambda \geq 5\sqrt{n}$, the improvement could be in the order of $\sqrt{\frac{\log n}{n}}$.

4 Kernelization Algorithms for Line Cover

In this section, we present a randomized Monte Carlo kernelization algorithm for Line Cover that employs **Alg-RichLines** developed in the previous section. We also show how the tools developed in this section can be used to obtain a deterministic kernelization algorithm for Line Cover that employs Guibas et al.'s algorithm [25]. Both algorithms improve the running time of existing kernelization algorithms for Line Cover. Moreover, we will show in Section 5 that the running time of our randomized algorithm comes close to the lower bound that we derive on the time complexity of kernelization algorithms for Line Cover in the algebraic computation trees model. The majority of this section is dedicated to proving the following theorem:

▶ **Theorem 9.** *There is a Monte Carlo randomized algorithm,* **Alg-Kernel***, that given an instance (S, k) of* Line Cover*, in time $\mathcal{O}(n\log k + k^2(\log^2 k)(\log\log k)^2)$, returns an instance (S', k') such that $|S'| \leq k^2$, and such that with probability at least $1 - \frac{2}{k^3}$, (S', k') is a kernel of (S, k). More specifically: (1) if (S, k) is a yes-instance of* Line Cover*, then with probability at least $1 - \frac{2}{k^3}$, (S', k') is a yes-instance of* Line Cover*; and (2) if (S, k) is a no-instance of* Line Cover *then (S', k') is a no-instance of* Line Cover*. The space complexity of this algorithm is $\mathcal{O}(k^2\log^2 k)$.*

Let (S, k) be an instance of Line Cover. We say that a line l is *saturated* w.r.t. S if it is $(k + 1)$-rich for S; denote by the *saturation* of a saturated line l the number of points on l. A line l is *unsaturated* w.r.t. S if it is not saturated. We start by giving an intuitive explanation of the results leading to the kernelization algorithm **Alg-Kernel**.

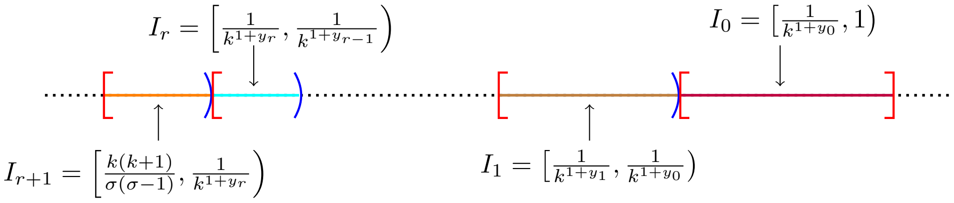
The kernelization algorithm processes the set S of points in "batches" of roughly $2k^2$ uncovered points each, and for each batch S', computes the saturated lines induced by S' and adds them to the (partial) solution. Since processing each batch should result in computing at least one saturated line – assuming a yes-instance, the above process iterates at most k times. The main task becomes to compute the saturated lines induced by a batch efficiently. One straightforward idea is to invoke **Alg-RichLines** directly with $\lambda = k + 1$, which, w.h.p., computes all the saturated lines in S'. The drawback is that **Alg-RichLines** takes time $\mathcal{O}(k^2 \log^2 k)$ per batch, and may result in a single saturated line, and hence in an overall running time of $\mathcal{O}(n \log k + k^3 \log^2 k)$ for the kernelization algorithm.

The main technical contributions of this section lie in devising a more efficient implementation of the above kernelization scheme. The improved scheme rests on two key observations: (1) the running time of **Alg-RichLines** decreases as the saturation threshold (i.e., λ) of the saturated lines sought increases; and (2) assuming that a subset of the batch S' needs to be covered only by saturated lines, then for any $\lambda < \lambda'$, it requires more (saturated) lines of saturation λ to cover S' than lines of saturation λ'.

Based on the above observations, we design an algorithm **Alg-SaturatedLines** that intuitively works as follows. We first partition the saturation range into intervals, thus defining a spectrum of saturation levels. Then **Alg-SaturatedLines** calls **Alg-RichLines** starting with the highest saturation threshold (i.e., starting with a value of λ defining the highest saturation interval in the spectrum), and iteratively decreasing the saturation threshold until either: (1) the saturated lines computed cover "enough" points of the batch S', or (2) the total number of saturated lines computed for the batch S' is "large enough", thus making enough progress towards computing the k lines in the line cover of (S, k).

The above scheme enables us to amortize the running time of **Alg-SaturatedLines** over the number of saturated lines it computes. The main kernelization algorithm, **Alg-Kernel**, then calls **Alg-SaturatedLines** on each batch of $2k^2$ uncovered points. As we show in the analysis, the above scheme enables a win/win situation, yielding an overall running time of $\mathcal{O}(n \log k + k^2(\log^2 k)(\log\log k)^2)$.

We also show that, if instead of using the randomized Monte Carlo algorithm **Alg-RichLines** to compute the saturated lines we use the deterministic algorithm of Guibas et al. [25], the above scheme yields a deterministic kernelization algorithm for LINE COVER that runs in time $\mathcal{O}(n \log k + k^3(\log^3 k)\sqrt{\log\log k})$ and computes a kernel of size at most k^2.



$I_r = \left[\frac{1}{k^{1+y_r}}, \frac{1}{k^{1+y_{r-1}}}\right)$
$I_0 = \left[\frac{1}{k^{1+y_0}}, 1\right)$
$I_{r+1} = \left[\frac{k(k+1)}{\sigma(\sigma-1)}, \frac{1}{k^{1+y_r}}\right)$
$I_1 = \left[\frac{1}{k^{1+y_1}}, \frac{1}{k^{1+y_0}}\right)$

Figure 2 Illustration for the definition of the intervals $I_0, \ldots, I_{r+1}$.

We now give an intuitive low-rigor description of the technical results leading to the kernelization algorithm. Lemma 10 is a combinatorial result showing that either the saturated lines belonging to the highest interval in the saturation spectrum cover enough points of the batch S', or there is a saturation interval in the spectrum containing a "large enough" number of lines. Lemma 10 is then employed by Lemma 11 to show that, w.h.p., **Alg-SaturatedLines** returns a set of saturated lines that either covers enough points of the batch S', or contains a "large enough" number of saturated lines. We employ Lemma 10 and use amortized analysis to upper bound the running time of **Alg-SaturatedLines** w.r.t. the number of saturated lines computed by this algorithm, which we subsequently use to upper bound the running time of **Alg-SaturatedLines** in Lemma 13. Finally, Theorem 9 employs the above results to prove the correctness of **Alg-Kernel** and upper bounds its time and space complexity. We now proceed to the details.

In what follows let $\sigma = 2k^2$, and let $S' \subseteq S$ be a subset of points such that $|S'| = \sigma$. We want to identify a subset of saturated lines w.r.t. S'. We define the following notations. Let $\epsilon = \frac{\ln\ln\ln k}{\ln k}$. For $i \in \mathbb{N}$, let $y_i = 1 - \frac{\ln\ln k}{\ln k} - \frac{\ln\ln\ln k}{\ln k} + i\epsilon$. Let r be the minimum integer such that $k^{y_r} \geq \frac{k}{(\ln\ln k)^2}$, and note that $r = \mathcal{O}(\ln\ln k)$. Note that we have $y_0 < y_1 < \cdots < y_r$.

We define a sequence of intervals $I_0, \ldots, I_{r+1}$ as follows: $I_0 = [\frac{1}{k^{1+y_0}}, 1] = [\frac{\ln k(\ln\ln k)}{k^2}, 1]$, $I_i = [\frac{1}{k^{1+y_i}}, \frac{1}{k^{1+y_{i-1}}})$, for $i = 1, 2, \ldots, r$, and $I_{r+1} = [\frac{(k+1)k}{\sigma(\sigma-1)}, \frac{1}{k^{1+y_r}})$. Observe that the intervals $I_0, \ldots, I_{r+1}$ are mutually disjoint, and partition the interval $[\frac{(k+1)k}{\sigma(\sigma-1)}, 1]$. It is easy to verify that the lengths of the intervals $I_1, \ldots, I_r$ are decreasing. See Figure 2 for illustration.

Suppose that there are h saturated lines $l_1, \ldots, l_h$ w.r.t. S'. Denote by s_i the number of points in S' covered by l_i, for $i \in [h]$. Let $\rho_i = \frac{s_i(s_i-1)}{\sigma(\sigma-1)}$, and note that ρ_i belongs to one of the intervals $I_0, \ldots, I_{r+1}$. We partition the h saturated lines into at most $r+2$ groups, $H'_0, \ldots, H'_{r+1}$, where H'_i, for $i = 0, \ldots, r+1$, consists of every saturated line l_j, $j \in [h]$, such that $\rho_j \in I_i$. Clearly, it follows that $H'_0, \ldots, H'_{r+1}$ is indeed a partitioning of $\{l_1, \ldots, l_h\}$.

Consider **Alg-SaturatedLines** for computing the saturated lines w.r.t. S':

Algorithm 2 : Alg-SaturatedLines(S', k, r) – A randomized algorithm for computing saturated lines w.r.t. S'.

```
Input: $S'$, where $|S'| = \sigma = 2k^2$; $k \in \mathbb{N}$; and integer $r$ as defined before
Output: A set of points $S''$ and a set of saturated lines $L'$
for $(i = 0; i \leq r+1; i++)$ do
    if $i \leq r$ then let $L' = $ Alg-RichLines$(S', \sigma k^{-(1+y_i)/2})$;
    else let $L' = $ Alg-RichLines$(S', k+1)$;
    compute the set $S'' \subseteq S'$ not covered by $L'$;
    if $i = 0$ and $L'$ covers at least $k^2/3$ points then return $(S'', L')$;
    else if $i \leq r$ and $|L'| \geq \frac{1}{12r}k^{(1+y_{i-1})/2}$ then return $(S'', L')$;
    else if $i = r+1$ and $|L'| \geq \frac{1}{12}k^{(1+y_r)/2}$ then return $(S'', L')$;
return $(S', \emptyset)$;
```

Now we are ready to present the kernelization algorithm, **Alg-Kernel**, for Line Cover.

The kernelization algorithm works by computing w.h.p. the set H of saturated lines in S and removing all points covered by these lines. Observe that, any set of more than k^2 points that can be covered by at most k lines must contain at least one saturated line. During the execution of the algorithm, the set S', which will eventually contain the kernel, contains a subset of points in S. We start by initializing S' to the empty set, and order the points in S arbitrarily. We repeatedly add the next point in S (w.r.t. the defined order) to S' until either $|S'| = 2k^2$, or no points are left in S. Afterwards, the algorithm distinguishes two cases.

If $|S'| = 2k^2$, the algorithm calls **Alg-SaturatedLines** to compute a subset of the saturated lines w.r.t. S'. **Alg-SaturatedLines** may not compute all the saturated lines in S', and rather acts as a "filtering algorithm". This algorithm either computes a subset of saturated lines that cover at least $k^2/3$ many points in S' "efficiently", that is more efficiently than **Alg-RichLines**, which w.h.p. computes all the saturated lines in S'; or computes a "large" set of saturated lines (a little bit less efficiently than **Alg-RichLines**), thus decreasing the parameter k significantly (and hence the overall execution of the algorithm).

If $k^2 < |S'| < 2k^2$, no more points are left in S to consider. **Alg-RichLines** is called at most once to compute w.h.p. all the remaining saturated lines w.r.t. S' to return the kernel.

We now proceed to prove the correctness and analyzing the complexity of **Alg-Kernel**.

▶ **Lemma 10.** *Given a set S' of σ points and a parameter k, if S' can be covered by at most k lines then one of the following conditions must hold:*

(1) *H'_0 covers at least $\frac{\sigma-k^2}{3}$ points;*

(2) *$|H'_i| \geq (\frac{\sigma-k^2}{6r\sigma}) \cdot k^{(1+y_{i-1})/2}$ for some $i \in [r]$; or*

(3) *$|H'_{r+1}| \geq (\frac{\sigma-k^2}{6\sigma}) \cdot k^{(1+y_r)/2}$.*

Algorithm 3 : **Alg-Kernel**(S, k)–A randomized kernelization algorithm for Line Cover.

```
Input: $S = \{q_1, \dots, q_n\}$; $k \in \mathbb{N}$.
Output: an instance $(S', k')$ of Line Cover.
if $k \leq 15$ then return the instance $(S', k')$ described in Lemma 12;
$H = \emptyset$; $S' = \emptyset$; $i = 1$;
construct a search structure $\Gamma_H$ for the lines in $H$ and set $\Gamma_H = \emptyset$;
while $|H| \leq k$ do
    while $|S'| < 2k^2$ and $i \leq n$ do
        if $q_i$ is not covered by $H$ then add $q_i$ to $S'$ and set $i = i + 1$;
    if $|S'| = 2k^2$ then
        let $(S', L') =$ Alg-SaturatedLines$(S', k, r)$;
        if $L' = \emptyset$ then return a (trivial) no-instance $(S', k')$;
        $H = H \cup L'$; update $\Gamma_H$ for $H$;
    else
        if $|S'| > k^2$ then
            $L' =$ Alg-RichLines$(S', k + 1)$; $H = H \cup L'$;
            update $\Gamma_H$ for $H$; remove the points from $S'$ covered by $L'$;
            if $|H| > k$ or $|S'| > k^2$ then return a (trivial) no-instance $(S', k')$;
        return $(S', k - |H|)$;
return a (trivial) no-instance $(S', k')$;
```

▶ **Lemma 11.** *Given a set S' of points and a parameter $k \geq 16$, let L' be the set of lines returned by* **Alg-SaturatedLines**(S', k, r)*. If S' can be covered with at most k lines, then with probability at least $1 - \frac{1}{k^4}$ one of the following holds:*

(1) *L' covers at least $\frac{k^2}{3}$ points;*

(2) *$|L'| \geq \frac{1}{12r} k^{(1+y_{i-1})/2}$ for some $i \in [r]$; or*

(3) *$|L'| \geq \frac{1}{12} k^{(1+y_r)/2}$.*

One technicality ensues from the definition of the saturation intervals. Since this definition entails using the term $\ln \ln \ln k$, $\ln \ln \ln k$ must be positive, and hence $k \geq 16 > e^e$. This forces a separate treatment of instances in which $k \leq 15$. Since $k = \mathcal{O}(1)$, we could opt to use a brute-force algorithm in this case, or an FPT-algorithm, but those would result in a polynomial running time of a higher degree than what is desired for our purpose. Instead, we provide an efficient linear-time algorithm for this special case in the following lemma:

▶ **Lemma 12.** *Given an instance (S, k) of* Line Cover*, where $|S| = n$ and $k \leq 15$, there is an algorithm that computes in $\mathcal{O}(n)$ time and $\mathcal{O}(1)$ space a kernel (S', k') for (S, k) such that $|S'| \leq k^2$.*

▶ **Lemma 13.** *Given an instance (S, k) of* Line Cover*, where $|S| = n$,* **Alg-Kernel** *runs in time $\mathcal{O}(n \log k + k^2 (\log^2 k)(\log \log k)^2)$ and space $\mathcal{O}(k^2 \log^2 k)$.*

Proof of Theorem 9 (stated at the beginning of this section). The time and space complexity of the algorithm follow from Lemmas 12 and 13. We prove its correctness next. The correctness of Step 1 was proved separately in Lemma 12, so we may assume that $k \geq 16$.

Suppose that (S, k) is a no-instance of Line Cover. Observe that whenever the algorithm includes a subset L' of lines into the solution H (in Steps 10 and 13) (and updates S'), then the lines in L' are saturated lines, and hence, must be part of *every* solution to the instance (S, k). Therefore, either the algorithm returns an instance in Step 16 that must be a no-instance by the above observation, or returns a (trivial) no-instance in Step 9, 15, or 17. It follows from above that if (S, k) is a no-instance of Line Cover, then **Alg-Kernel** returns a no-instance (S', k'). This proves part (2) of the theorem.

Suppose now that (S, k) is a yes-instance of Line Cover, and hence, that S can be covered by at most k lines. By Step 9, if $L' = \emptyset$, then the algorithm will stop. Thus, Steps 7–10 will be executed at most $k + 1$ times. Consider a single execution of Steps 7–10. By Lemma 11, if S' can be covered with at most k saturated lines, then, with probability at least $1 - \frac{1}{k^4}$, **Alg-SaturatedLines**(S', k, r) returns a non-empty set L'. That is to say, **Alg-SaturatedLines**(S', k, r) fails with probability at most $\frac{1}{k^4}$. By the union bound, **Alg-Kernel**(S, k) fails during the execution of Steps 7–10 with probability at most $\frac{k+1}{k^4}$. At Step 13, by Theorem 8, with probability at least $1 - \frac{3}{|S'|^2} > 1 - \frac{3}{k^4}$, **Alg-RichLines**$(S', k)$ finds all the saturated lines in S'. After that, we have $|S'| \leq k^2$. By the union bound, with probability at least $1 - \frac{k+1}{k^4} - \frac{3}{k^4} > 1 - \frac{2}{k^3}$ (since $k \geq 16$), **Alg-Kernel**(S, k) returns a kernel (S', k') of (S, k) satisfying $|S'| \leq k^2$. This proves part (1) of the theorem. ◀

We conclude this section by giving a deterministic kernelization algorithm for Line Cover. Recall that **Alg-RichLines** is a randomized algorithm for computing all λ-rich lines and that Guibas et al.'s algorithm [25] is a deterministic algorithm for the same purpose. We can replace **Alg-RichLines** with Guibas et al.'s algorithm [25] in the algorithms **Alg-SaturatedLines** and **Alg-Kernel** to obtain a deterministic kernelization algorithm from **Alg-Kernel** after this replacement. We can optimize the running time of this deterministic algorithm by fine-tuning the lengths of the defined intervals $I_0, \ldots, I_{r+1}$.

▶ **Theorem 14.** *There is a deterministic kernelization algorithm for* Line Cover *that, given an instance* (S, k) *of* Line Cover*, where* $|S| = n$*, the algorithm runs in time* $\mathcal{O}(n \log k + k^3 (\log^3 k) \sqrt{\log \log k})$ *and computes a kernel* (S', k') *such that* $|S'| \leq k^2$*.*

5 Lower Bounds

In this section, we establish time-complexity lower-bound results for Line Cover and Rich Lines in the algebraic computation trees model [6]. The algebraic computation trees model is a more powerful model than the real-RAM model [19], which is the model of computation that is most commonly used to analyze geometric algorithms [38]. The lower-bound results we derive in the algebraic computation trees model apply to the real RAM model as well; for more details see [19].

5.1 Line Cover

In order to derive lower bounds on the time complexity of kernelization algorithms for Line Cover, we combine a lower-bound result by Grantson and Levcopoulos [23] on the time complexity of Line Cover with a result that we prove below connecting the time complexity for solving Line Cover to its kernelization time complexity. We remark that, since Line

Cover is NP-hard [33] when the parameter k is unbounded, Grantson and Levcopoulos' [23] time complexity lower-bound result for Line Cover is interesting only when k is "small" relative to the input size, and should be read this way.

▶ **Theorem 15** (Grantson and Levcopoulos [23]). *There exists a constant $c > 0$ such that, for every positive $n, k \in \mathbb{N}$ satisfying $k = \mathcal{O}(\sqrt{n})$,* Line Cover *requires time at least $c \cdot n \log k$ in the algebraic computation trees model.*

We now exploit a folklore connection between kernelization and FPT [13, 14] to translate the above time-complexity lower-bound result into a kernelization time-complexity lower-bound result.

▶ **Theorem 16.** *Let Q be a parameterized problem in* NP*. For any proper complexity function h, Q has a kernelization algorithm of running time $\mathcal{O}(h(|x|, k))$, where (x, k) is the input instance to Q, if and only if Q can be solved in time $\mathcal{O}(h(|x|, k) + g(k))$ for some proper complexity function $g(k)$.*

The corollary below follows from Theorem 15 and Theorem 16 above:

▶ **Corollary 17.** *There exists a constant $c > 0$ such that the running time of any kernelization algorithm for* Line Cover *in the algebraic computation trees model is at least $cn \log k$.*

▶ Remark 18. The above corollary implies that one cannot asymptotically improve on either of the two factors n or $\log k$ in the term $n \log k$. This rules out, for instance, the possibility of a kernelization algorithm that runs in (linear) $O(n)$ time or in $\mathcal{O}(n \log \log k)$ time.

5.2 Rich Lines

In this subsection, we derive lower-bound results on the time complexity of Rich Lines in the algebraic computation trees model using Ben-Or's framework [4]. Consider the variant of the Element Distinctness problem [4]:

> Multiset Subset Distinctness
> Given a multi-set $A = \{a_1, a_2, \ldots, a_n\}$ and a positive integer λ, decide whether A can be partitioned into n/λ multi-subsets $A_1, A_2, \ldots, A_{n/\lambda}$, such that each subset A_i, where $i \in [n/\lambda]$, contains exactly λ identical elements, and no two (distinct) multi-subsets contain identical elements.

▶ **Theorem 19.** *There exists a constant $c > 0$ such that, for every positive $n, \lambda \in \mathbb{N}$ such that λ divides n,* Multiset Subset Distinctness *requires time at least $c \cdot n \log(\frac{n}{\lambda})$ in the algebraic computation trees model.*

Proof. For any fixed n and λ, the instance (A, λ), where $A = (a_1, \ldots, a_n)$, is represented as the point $(a_1, \ldots, a_n, \lambda)$ in the $(n+1)$-dimensional Euclidean space R^{n+1}. Denote by W_λ^{n+1} the set of points in R^{n+1} that corresponds to the set of yes-instances of Multiset Subset Distinctness. By Ben-Or's results [4, §4], it suffices to show that the number of connected components of W_λ^{n+1} is at least $\binom{n}{\lambda,\lambda,\ldots,\lambda} = \Theta(\frac{\sqrt{2\pi n}(n/e)^n}{(\sqrt{2\pi\lambda}(\lambda/e)^\lambda)^{n/\lambda}})$ [22, §9.6], as this would show that the depth of any algebraic computation tree for Multiset Subset Distinctness is at least $\Omega(\log \binom{n}{\lambda,\lambda,\ldots,\lambda}) = \Omega(n \log(\frac{n}{\lambda}))$.

Each yes-instance (A, λ) of Multiset Subset Distinctness corresponds to a mapping f from $[n] \to [n/\lambda]$ such that $f(i) < f(j)$ if and only if $a_i < a_j$, and such that $f(i) = f(j)$ if and only if $a_i = a_j$, and such that for each $j \in [n/\lambda]$: $|\{i \in [n] \mid f(i) = j\}| = \lambda$. It is easy to see that the number of such functions f is $\binom{n}{\lambda,\lambda,\ldots,\lambda}$. For each such function f, let W_f be the set of yes-instances corresponding to f, and let $\mathcal{W}$ be the set of all subsets W_f. It is easy to verify that the sets W_f in $\mathcal{W}$ partition W_λ^{n+1}, and that W_f is a connected region/subset in $\mathbb{R}^{n+1}$, as it is the intersection of hyperplanes with a convex set/region.

We prove that, for any two different functions f and f', W_f and $W_{f'}$ belong to two different connected component of W_λ^{n+1}. Assume to the contrary that a point $p \in W_f$ and a point $p' \in W_{f'}$ are in the same connected component of the set W_λ^{n+1}. Then there is a path Π in W_λ^{n+1} from p to p'. This path Π can be given in the parametric form as:

$$\Pi : \pi(t) = (a_1(t), a_2(t), \ldots, a_n(t), \lambda), 0 \leq t \leq 1,$$

where $\pi(0) = p$, $\pi(1) = p'$, and each $a_i(t)$, $i \in [n]$ is a continuous function of t. For an interval $I \subseteq [0,1]$, denote by $\pi(I) = \{\pi(t) \mid t \in I\}$.

Suppose first that, for each $t \in [0,1]$, there is an open interval I_t containing t such that all points in $\pi(I_t)$ are in the same subset of $\mathcal{W}$. Then by the Heine-Borel Theorem [39], we can find a finite set of open intervals covering $[0,1]$ such that for each such open interval I_t, all points in $\pi(I_t)$ are in the same subset of $\mathcal{W}$. This implies that all points on the path Π are in the same subset of $\mathcal{W}$, contradicting the fact that the subsets W_f and $W_{f'}$ are disjoint.

Suppose now that there exists a $t_0 \in [0,1]$, where $\pi(t_0)$ is in some W_{f_1}, such that for every open interval I containing t_0, $\pi(I)$ contains a point not in W_{f_1}. Since $\mathcal{W}$ is finite, we can construct a sequence $(t)_i$ in $[0,1]$ converging to t_0, and such that, for each i, $\pi(t_i)$ belongs to the *same* set $W_{f_2} \in \mathcal{W}$, where $f_1 \neq f_2$. Since $f_1 \neq f_2$, there exist indices z_1 and z_2 such that $z_1 \neq z_2$, $f_1(z_1) < f_1(z_2)$ and $f_2(z_1) > f_2(z_2)$. Consider the sequence of points

$$\pi(t_r) = (a_1(t_r), a_2(t_r), \ldots, a_n(t_r), \lambda), \text{ for } r \geq 1.$$

Since $\pi(t_r)$ approaches $\pi(t_0)$ as $t_r \to t_0$, we must have

$$|a_{z_1}(t_r) - a_{z_1}(t_0)| + |a_{z_2}(t_r) - a_{z_2}(t_0)| \to 0, \tag{1}$$

as $t_r \to t_0$. Recall that $f_1(z_1) < f_1(z_2)$ and $f_2(z_1) > f_2(z_2)$, and hence, $a_{z_1}(t_0) < a_{z_2}(t_0)$ and $a_{z_1}(t_r) > a_{z_2}(t_r)$. It follows that:

$$\begin{aligned}
&|a_{z_1}(t_r) - a_{z_1}(t_0)| + |a_{z_2}(t_r) - a_{z_2}(t_0)| && (2)\\
\geq\ &|(a_{z_1}(t_r) - a_{z_1}(t_0)) - (a_{z_2}(t_r) - a_{z_2}(t_0))| && (3)\\
\geq\ &|(a_{z_2}(t_0) - a_{z_1}(t_0)) + (a_{z_1}(t_r) - a_{z_2}(t_r))| && (4)\\
\geq\ &|(a_{z_2}(t_0) - a_{z_1}(t_0))|. && (5)
\end{aligned}$$

Observing that $a_{z_1}(t_0)$ and $a_{z_2}(t_0)$ are fixed, inequality (5) contradicts (1). This completes the proof. ◀

Now, we can prove a time lower bound $\Omega(n \log \frac{n}{\lambda})$ for the Rich Lines problem via a reduction from Multiset Subset Distinctness problem.

▶ **Theorem 20.** *There exists a constant $c_0 > 0$ such that, for every positive $n, \lambda \in \mathbb{N}$,* Rich Lines *requires time at least $c_0 \cdot n \log(\frac{n}{\lambda})$ in the algebraic computation trees model.*

Proof. We prove the theorem via a Turing-reduction $\mathcal{T}$ from the Multiset Subset Distinctness problem. The theorem would then follow from Theorem 19. We first present the reduction.

Given an instance $(A, \lambda) = (a_1, a_2, \ldots, a_n, \lambda)$ of Multiset Subset Distinctness, we construct the instance (P, λ) of Rich Lines, where $P = \{(a_i, i) \mid a_i \in A\}$. Note that (P, λ) can be constructed in $\mathcal{O}(n)$ time. Observe that (A, λ) is a yes-instance of Multiset Subset

Distinctness if and only if there are n/λ vertical lines that each covers exactly λ points of P. We can solve (P, λ) to find the set L of lines induced by P that each covers at least λ points. Then, we compute the subset V of vertical lines in L and accept (A, λ) if and only if $|V| = n/\lambda$. Let $t(n, \lambda)$ be the time needed to perform this reduction $\mathcal{T}$.

Now to prove the theorem, we proceed by contradiction. Suppose that no such constant c_0 exists, and let c be the universal constant in Theorem 19. Then, for every constant $c' > 0$, there exist $n, \lambda \in \mathbb{N}$ such that, for *all* input instances of size n and parameter λ, Rich Lines can be solved in time less than $c' \cdot n \log(\frac{n}{\lambda})$. We observe that, under this assumption, the number of lines in the solution to each of these instances must be less than $c' \cdot n \log(\frac{n}{\lambda})$, otherwise, the running time for solving the instance would necessarily exceed $c' \cdot n \log(\frac{n}{\lambda})$. It is not difficult to see that we can choose a constant $c' > 0$ and $n, \lambda \in \mathbb{N}$ such that for the specific function $t(n, \lambda)$, where $t(n, \lambda)$ is running time of the reduction $\mathcal{T}$ given above, we have $t(n, \lambda) + c' \cdot n \log(\frac{n}{\lambda}) < c \cdot n \log(\frac{n}{\lambda})$. Let n, λ be the values chosen accordingly.

Assume first that λ divides n, and we explain below how the proof can be modified to lift this assumption. Given an instance $(A, \lambda) = (a_1, a_2, \dots, a_n, \lambda)$ of Multiset Subset Distinctness, where A has n elements, we reduce (A, λ) via reduction $\mathcal{T}$ to an instance (P, λ) of Rich Lines and solve (P, λ) to obtain a solution to (A, λ) in time less than $cn \log(\frac{n}{\lambda})$, contradicting Theorem 19.

In the case where λ does not divide n, let $n = r \cdot \lambda + s$, where $s < \lambda$, and let $n' = r \cdot \lambda$. Observe that the lower bound for Multiset Subset Distinctness established in Theorem 19 holds for the values n', λ (since λ divides n'). Given an instance $(A', \lambda) = (a_1, a_2, \dots, a_{n'}, \lambda)$ of Multiset Subset Distinctness, we construct the instance (P, λ) of Rich Lines, where $P = P' \cup S$, and $P' = \{(a_i, i) \mid a_i \in A'\}$. The set S contains precisely $s < \lambda$ points and is constructed as follows. We find the smallest element $a_{min} \in A'$, and choose a number $x < a_{min}$. Define $S = \{(x, j) \mid j \in [s]\}$. It is easy to verify that (A', λ) is a yes-instance of Multiset Subset Distinctness if and only if the number of vertical lines, each containing at least λ points of P, is n'/λ. Hence, we can decide (A', λ) as explained in the first case above. Note that all the steps involved in the construction of (P, λ), including the computation of the number x, can be carried out in linear time. Since the constant c' can be chosen to be arbitrary small, it is not difficult to see that we can choose c' and the values n, λ such that the running time of the above reduction is less than $c \cdot n' \log(\frac{n'}{\lambda})$, again contradicting Theorem 19. Note also that all the operations involved in the above reduction can be equivalently modeled in the algebraic computation trees model [19]. This completes the proof. ◀

6 Conclusion

Several interesting questions ensue from our work. First, many of the previous algorithms for Rich Lines and Line Cover can be lifted to higher dimensions (e.g., see [25, 31, 41]). We believe that it is possible to lift the results in this paper to higher dimensions as well (where the covering objects are hyperplanes). Second, most of the algorithms we presented are randomized Monte Carlo algorithms. It is interesting to investigate if these algorithms can be derandomized without trading off their performance guarantees by much. Finally, it is interesting to see if the sampling and optimization techniques developed in this paper can be applied to other related problems in computational geometry. We leave all the above questions as directions for future research.

References

1 P. Afshani, E. Berglin, I. van Duijn, and J. Nielsen. Applications of incidence bounds in point covering problems. In *Proc. 32nd International Symposium on Computational Geometry (SoCG 2016),* Article No. 60, pages 1–15, 2016.

2 P. K. Agarwal and S. Sen. Randomized algorithms for geometric optimization problems. In *Handbook of Randomized Computation*, pages 151–201. Kluwer Academic Press, 2001.

3 J. Alman, M. Mnich, and V. V. Williams. Dynamic parameterized problems and algorithms. In *Proc. 44th International Colloquium on Automata, Languages and Programming (ICALP 2017),* Article No. 41, pages 1–16, 2017.

4 M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th ACM Symposium on Theory of Computing (STOC 1983)*, pages 80–86, 1983.

5 H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14:263–279, 1995.

6 P. Bürgisser, M. Clausen, and M. Shokrollahi. *Algebraic Complexity Theory.* Springer, Berlin, 1997.

7 C. Cao. *Study on two optimization problems: Line cover and maximum genus embedding.* PhD thesis, Texas A&M University, 2012.

8 R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, and M. Monemizadeh. New streaming algorithms for parameterized maximal matching and beyond. In *Proc. 27th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA 2015)*, pages 56–58, 2015.

9 K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

10 K. L. Clarkson. Randomized geometric algorithms. In *Computing in Euclidean Geometry*, volume 1, pages 117–162. World Scientific, 1992.

11 K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop Algorithms Data Struct*, pages 246–252, 1993.

12 K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

13 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms.* Springer, 2015.

14 R. Downey and M. Fellows. *Fundamentals of Parameterized Complexity.* Springer, New York, 2013.

15 J. Erickson. New lower bounds for Hopcroft's problem. *Discrete & Computational Geometry*, 16:389–418, 1996.

16 V. Estivill-Castro, A. Heednacram, and F. Suraweera. Reduction rules deliver efficient FPT-algorithms for covering points with lines. *Journal of Experimental Algorithmics*, 14:1–7, 2010.

17 V. Estivill-Castro, A. Heednacram, and F. Suraweera. FPT-algorithms for minimum-bends tours. *International Journal of Computational Geometry & Applications*, 21(2):189–213, 2011.

18 J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, Berlin, 2010.

19 H. Fournier and A. Vigneron. A tight lower bound for computing the diameter of a 3D convex polytope. *Algorithmica*, 49:245–257, 2007.

20 D. A. Freedman. *Statistical Model: Theory and Practice.* Cambridge University Press, 2009.

21 V. Froese, I. A. Kanj, A. Nichterlein, and R. Niedermeier. Finding points in general position. *International Journal of Computational Geometry and Applications*, 27(4):277–296, 2017.

22 R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics: A foundation for computer science.* Addison-Wesley, Reading, MA, 1989.

23 M. Grantson and C. Levcopoulos. Covering a set of points with a minimum number of lines. In *Proc. 6th Italian Conference on Algorithms and Complexity (CIAC 2006)*, pages 6–17. Lecture Notes in Computer Science 3998, 2006.

24 J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of patterns in 2D trajectories of moving points. *Geoinformatica*, 11(2):195–215, 2007.

25 L. J. Guibas, M. H. Overmars, and J. M. Robert. The exact fitting problem in higher dimensions. *Computational geometry*, 6:215–230, 1996.

26 M. Houle, H. Imai, K. Imai, J. Robert, and P. Yamamoto. Orthogonal weighted linear L_1 and L_∞ approximation and applications. *Discrete Applied Mathematics*, 43(3):217–232, 1993.

27 M. Houle and T. Toussaint. Computing the width of a set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):761–765, 1988.

28 D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

29 S. Kratsch, G. Philip, and S. Ray. Point line cover: the easy kernel is essentially tight. *ACM Transactions on Algorithms*, 12:3, 2016.

30 V. A. Kumar, S. Arya, and H. H. Ramesh. Hardness of set cover with intersection 1. In *Proc. 27th International Colloquium on Automata, Languages, and Programming (ICALP 2000)*, pages 624–635, 2000.

31 S. Langerman and P. Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005.

32 J. Matousek. Range searching with efficient hierarchical cutting. *Discrete Computational Geometry*, 10:157–182, 1993.

33 N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.

34 M. Mitzenmacher and E. Upfal. *Probability and Computing.* Cambridge University Press, 2nd edition, 2017.

35 M. Mnich. Big data algorithms beyond machine learning. *Künstliche Intelligencz*, 32:9–17, 2018.

36 R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

37 J. Pach, R. Radoicic, G. Tardos, and G. Toth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete & Computational Geometry*, 36(4):527–552, 2006.

38 F. Preparata and I. Shamos. *Computational Geometry: An Introduction.* 2nd edn. Texts and Monographs in Computer Science. Springer, New York, 1985.

39 W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, 3rd edition, 1976.

40 E. Szemerédi and W. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3(3-4):381–392, 1983.

41 J. Wang, W. Li, and J. Chen. A parameterized algorithm for the hyperplane-cover problem. *Theoretical Computer Science*, 411:4005–4009, 2010.

Towards Uniform Certification in QBF

Leroy Chew
TU Wien, Austria

Friedrich Slivovsky
TU Wien, Austria

Abstract

We pioneer a new technique that allows us to prove a multitude of previously open simulations in QBF proof complexity. In particular, we show that extended QBF Frege p-simulates clausal proof systems such as IR-Calculus, IRM-Calculus, Long-Distance Q-Resolution, and Merge Resolution. These results are obtained by taking a technique of Beyersdorff et al. (JACM 2020) that turns strategy extraction into simulation and combining it with new local strategy extraction arguments.

This approach leads to simulations that are carried out mainly in propositional logic, with minimal use of the QBF rules. Our proofs therefore provide a new, largely propositional interpretation of the simulated systems. We argue that these results strengthen the case for uniform certification in QBF solving, since many QBF proof systems now fall into place underneath extended QBF Frege.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, Proof Complexity, Verification, Frege, Extended Frege, Strategy Extraction

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.22

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2021/144/`

Funding This work was supported by the Vienna Science and Technology Fund (WWTF) under grant ICT19-060.

1 Introduction

The problem of evaluating Quantified Boolean Formulas (QBF), an extension of propositional satisfiability (SAT), is a canonical PSPACE-complete problem [36, 1]. Many tasks in verification, synthesis and reasoning have succinct QBF encodings [35], making QBF a natural target logic for automated reasoning. As such, QBF has seen considerable interest from the SAT community, leading to the development of a variety of QBF solvers (e.g., [29, 19, 32, 20, 30]). The underlying algorithms are often highly nontrivial, and their implementation can lead to subtle bugs [9]. While formal verification of solvers is typically impractical, trust in a solver's output can be established by having it generate a proof trace that can be externally validated. This is already standard in SAT solving with the DRAT proof system [39], for which even formally verified checkers are available [15]. A key requirement for standard proof formats like DRAT is that they *simulate* all current and emerging proof techniques.

Currently, there is no decided-upon checking format for QBF proofs (although there have been some suggestions [22, 18]). The main challenge of finding such an universal format, is that QBF solvers are so radically different in their proof techniques, that each solver basically works in its own proof system. For instance, solvers based on CDCL and (some) clausal abstraction solvers can generate proofs in Q-resolution (Q-Res) [25] or long-distance Q-resolution (LD-Q-Res) [2], while the proof system underlying expansion based solvers combines instantiation of universally quantified variables with resolution (∀Exp+Res) [21]. Variants of the latter system have been considered: IR-calc (**I**nstantiation **R**esolution) admits instantiation with partial assignments, and IRM-calc (**I**nstantiation **R**esolution **M**erge) additionally incorporates elements of long-distance Q-resolution [7].

A universal checking format for QBF ought to simulate all of these systems. A good candidate for such a proof system has been identified in extended QBF Frege (eFrege + ∀red): Beyersdorff et al. showed [6] that a lower bound for eFrege + ∀red would not be possible without a major breakthrough.

In this work, we show that eFrege + ∀red does indeed p-simulate IR-calc, IRM-calc, Merge Resolution (M-Res) and LQU$^+$-Res (a generalisation of LD-Q-Res), thereby establishing eFrege + ∀red and any stronger system (e.g., QRAT [18] or G [28]) as potential universal checking formats in QBF. As corollaries, we obtain (known) simulations of ∀Exp+Res [23] and LD-Q-Res [24] by QRAT, as well as a (new) simulation of IR-calc by QRAT, answering a question recently posed by Chede and Shukla [10]. A simulation structure with many of the known QBF proof systems and our new results is given in Figure 1.

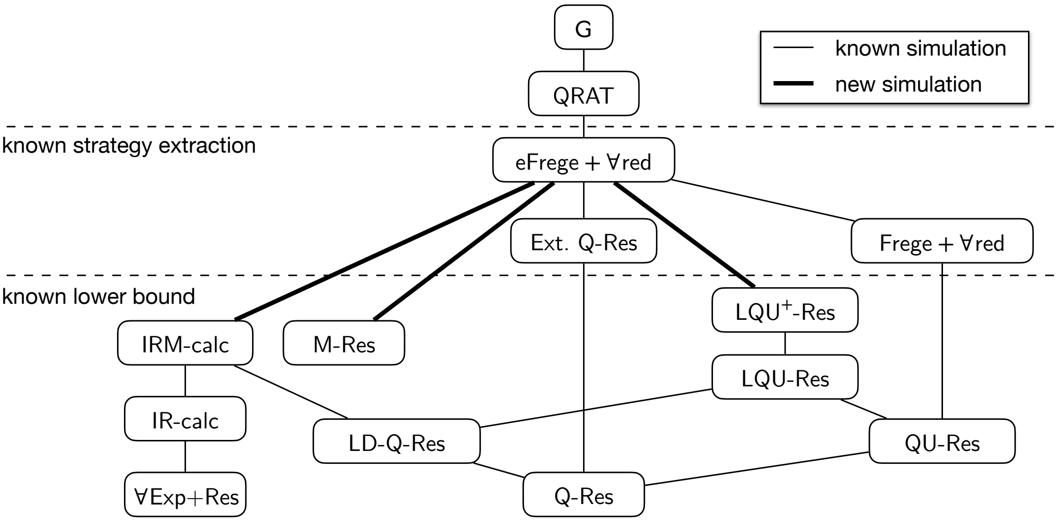


Figure 1 Hasse diagram for polynomial simulation order of QBF calculi [7, 3, 6, 18, 12, 2, 38, 13, 5]. In this diagram all proof systems below the first line are known to have strategy extraction, and all below the second line have an exponential lower bound. G and QRAT have strategy extraction if and only if P = PSPACE.

Our proofs crucially rely on a property of QBF proof systems known as strategy extraction. Here, "strategy" refers to winning strategies of a set of PSPACE two-player games (see Section 2 for more details) each of which corresponds exactly to some QBF. A proof system is said to have strategy extraction if a strategy for the two-player game associated with a QBF can be computed from a proof of the formula in polynomial time. Balabanov and Jiang discovered [2] that Q-Resolution admitted a form of strategy extraction where a circuit computing a winning strategy could be extracted in linear time from the proofs. Strategy extraction was subsequently proven for many QBF proof systems (cf. Figure 1): the expansion based systems ∀Exp+Res [7], IR-calc [7] and IRM-calc [7], Long-Distance Q-Resolution [16], including with dependency schemes [16], Merge Resolution [5], Relaxing Stratex [11] and C-Frege + ∀red systems including eFrege + ∀red [6]. Strategy extraction also gained notoriety because it became a method to show Q-resolution lower bounds [7]. Beyersdorff et al. [6, 8] generalised this approach to more powerful proof systems, allowing them to establish a tight correspondence between lower bounds for eFrege + ∀red and two major open problems in circuit complexity and propositional proof complexity: they showed that proving a lower bound for eFrege + ∀red is equivalent to either proving a lower bound for P/poly or a lower bound for propositional eFrege. Chew conjectured [12] that this meant that all the aforementioned proof systems that had strategy extraction were very likely to be simulated by eFrege + ∀red and showed an outline of how to use strategy extraction to obtain the corresponding simulations.

We follow this outline in proving simulations for multiple systems by eFrege + ∀red. While the strategy extraction for expansion based systems [7] has been known for a while using the technique from Goultiaeva et. al [17], there currently is no intuitive way to formalise this strategy extraction into a simulation proof. Here we specifically studied a new strategy extraction technique given by Schlaipfer et al. [34], that creates local strategies for each ∀Exp+Res line. Inductively, we can affirm each of these local strategies and prove the full strategy extraction this way. This local strategy extraction technique is based on arguments of Suda and Gleiss [37], which allow it to be generalised to the expansion based system IRM-calc. We thus manage to prove a simulation for ∀Exp+Res and generalise it to IR-calc and then to IRM-calc. We also show a much more straight-forward simulation of M-Res and an adaptation of the IRM-calc argument to LQU^{+}-Res.

The remainder of the paper is structured as follows. In Section 2 we go over general preliminaries and the definition of eFrege + ∀red. The remaining sections are each dedicated to simulations of different calculi by eFrege + ∀red. In Section 3 we begin with a simulation of M-Res as a relatively easy example. In Section 4 we show for expansion based systems, how both an interpretation by in propositional logic and a local strategy is possible and why that leads to a simulation by eFrege + ∀red. For IR-calc we state the essential lemmas of the proof and for IRM-calc we detail which modifications are needed. In Section 5 we study the strongest CDCL proof system LQU^{+}-Res and explain why it is also simulated by eFrege + ∀red, using a similar argument to IRM-calc.

2 Preliminaries

2.1 Quantified Boolean Formulas

A Quantified Boolean Formula (QBF) is a propositional formula augmented with Boolean quantifiers $\forall, \exists$ that range over the Boolean values $\bot, \top$ (the same as $0, 1$). Every propositional formula is already a QBF. Let ϕ be a QBF. The semantics of the quantifiers are that: $\forall x \phi(x) \equiv \phi(\bot) \wedge \phi(\top)$ and $\exists x \phi(x) \equiv \phi(\bot) \vee \phi(\top)$.

When investigating QBF in computer science we want to standardise the input formula. In a *prenex* QBF, all quantifiers appear outermost in a *(quantifier) prefix*, and are followed by a propositional formula, called the *matrix*. If every propositional variable of the matrix is bound by some quantifier in the prefix we say the QBF is a *closed* prenex QBF. We often want to standardise the propositional matrix, and so we can take the same approach as seen often in propositional logic. A *literal* is a propositional variable (x) or its negation ($\neg x$ or $\bar{x}$). A *clause* is a disjunction of literals. Since disjunction is idempotent, associative and commutative we can think of a clause simultaneously as a set of literals. The empty clause is just false. A *conjunctive normal form (CNF)* is a conjunction of clauses. Again, since conjunction is idempotent, associative and commutative a CNF can be seen as set of clauses. The empty CNF is true, and a CNF containing an empty clause is false. Every propositional formula has an equivalent formula in CNF, we therefore restrict our focus to closed *PCNF* QBFs, that is closed prenex QBFs with CNF matrices.

2.2 QBF Proof Systems

2.2.1 Proof Complexity

A proof system [14] is a polynomial-time checking function that checks that every proof maps to a valid theorem. Different proof systems have varying strengths, in one system a theorem may require very long proofs, in another the proofs could be considerably shorter. We use *proof complexity* to analyse the strength of proof systems [26]. A proof system is said to have an $\Omega(f(n))$-lower bound, if there is a family of theorems such that shortest proof (in number of symbols) of the family are bounded below by $\Omega(f(n))$ where n is the size (in number of symbols) of the theorem. Proof system p is said to *simulate* proof system q if there is a fixed polynomial $P(x)$ such that for every q-proof π of every theorem y there is a p-proof of y no bigger than $P(|\pi|)$ where $|\pi|$ denotes the size of π. A stricter condition, proof system p is said to p-simulate proof system q if there is a polynomial-time algorithm that takes q-proofs to p-proofs preserving the theorem.

2.2.2 Extended Frege+∀-Red

Frege systems are "text-book" style proof systems for propositional logic. They consist of a finite set of axioms and rules where any variable can be replaced by any formula (so each rule and axiom is actually a schema). A Frege system needs also to be sound and complete. Frege systems are incredibly powerful and can handle simple tautologies with ease. No lower bounds are known for Frege systems and all Frege systems are p-equivalent [14, 33]. For these reasons we can assume all Frege-systems can handle simple tautologies and syllogisms without going into details.

Extended Frege (eFrege) takes a Frege system and allows the introduction of new variables that do not appear in any previous line of the proof. These variables abbreviate formulas. The rule works by introducing the axiom of $v \leftrightarrow f$ for new variable v (not appearing in the formula f). Alternatively one can consider eFrege as the system where lines are circuits instead of formulas.

Extended Frege is a very powerful system, it was shown [27, 4] that any propositional proof system f can be simulated by $\mathsf{eFrege} + ||\phi||$ where ϕ is a polynomially recognisable axiom scheme. The QBF analogue is $\mathsf{eFrege} + \forall\mathsf{red}$, which adds the reduction rule to all existing eFrege rules [6]. $\mathsf{eFrege} + \forall\mathsf{red}$ is refutationally sound and complete for closed prenex QBFs. The reduction rules allows one to substitute a universal variable in a formula with 0 or with 1 as long as no other variable appearing in that formula is right of it in the prefix. Extension variables now must appear in the prefix and must be quantified right of the variables used to define it, we can consider them to be defined immediately right of these variables as there is no disadvantage to this.

2.3 QBF Strategies

With a closed prenex QBF $\Pi\phi$, the semantics of a QBF has an alternative definition in games. The two-player QBF game has an $\exists$-player and a $\forall$-player. The game is played in order of the prefix Π left-to-right, whoever's quantifier appears must assign the quantified variable to $\bot$ or $\top$. The existential player is trying to make the matrix ϕ become true. The universal player is trying to make the matrix become false. $\Pi\phi$ is true if and only if there winning strategy for the $\exists$ player. $\Pi\phi$ is false if and only if there winning strategy for the $\forall$ player.

A *strategy* for a false QBF is a set of functions f_u for each universal variable u on variables left of u in the prefix. In a *winning strategy* the propositional matrix must evaluate to false when every u is replaced by f_u. A QBF proof system has *strategy extraction* if there is a polynomial time program that takes in a refutation π of some QBF Ψ and outputs circuits that represent the functions of a winning strategy.

A *policy* is similarly defined as a strategy but with partial functions for each universal variables instead of a fully defined function.

3 Extended Frege+∀-Red p-simulates M-Res

In this section we show a first example of how the eFrege + ∀red simulation argument works in practice for systems that have strategy extraction. Merge resolution provides a straightforward example because the strategies themselves are very suitable to be managed in propositional logic. In later theorems where we simulate calculi like IR-calc and IRM-calc, representing strategies is much more of a challenge.

3.1 Merge Resolution

Merge resolution (M-Res) was first defined by Beyersdorff, Blinkhorn and Mahajan [5]. Its lines combines clausal information with a merge map, for each universal variable. Merge maps give a "local" strategy which when followed forces the clause to be true or the original CNF to be false.

3.1.1 Definition of Merge Resolution

Each line of an M-Res proof consists of a clause on existential variables and partial universal strategy functions for universal variables. These functions are represented by *merge maps*, which are defined as follows. For universal variable u, let E_u be the set of existential variables left of u in the prefix. A non-trivial merge map M^u_i is a collection of nodes in $[i]$, where the construction function $M^u_i(j)$ is either in $\{\bot, \top\}$ for leaf nodes or $E_u \times [j] \times [j]$ for internal nodes. The root $r(u, i)$ is the highest value of all the nodes M^u_i. The strategy function $h^u_{i,j} : \{0, 1\}^{E_u} \to \{0, 1\}$ maps assignments of existential variables E_u in the dependency set of u to a value for u. The function $h^u_{i,t}$ for leaf nodes t is simply the truth value $M^u_i(t)$. For internal nodes a with $M^u_i(a) = (y, b, c)$, we should interpret $h^u_{i,a}$ as "If y then $h^u_{i,b}$, else $h^u_{i,c}$" or $h^u_{i,a} = (y \wedge h^u_{i,b}) \vee (\neg y \wedge h^u_{i,c})$. In summary the merge map $M^u_i(j)$ is a representation of the strategy given by function $h^u_{i,r(u,i)}$.

The merge resolution proof system inevitably has merge maps for the same universal variable interact, and we have two kinds of relations on pairs of merge maps.

▶ **Definition 1.** *Merge maps M^u_j and M^u_k are said to be* consistent *if $M^u_j(i) = M^u_k(i)$ for each node i appearing in both M^u_j and M^u_k.*

▶ **Definition 2.** *Merge maps M^u_j and M^u_k are said to be isomorphic if is there exists a bijection f from the nodes of M^u_j to the nodes of M^u_k such that if $M^u_j(a) = (y, b, c)$ then $M^u_k(f(a)) = (y, f(b), f(c))$ and if $M^u_j(t) = p \in \{\bot, \top\}$ then $M^u_k(f(t)) = p$.*

With two merge maps M^u_j and M^u_k, we define two operations as follows:

- Select(M^u_j, M^u_k) returns M^u_j if M^u_k is trivial (representing a "don't care"), or M^u_j and M^u_k are isomorphic and returns M^u_k if M^u_j is trivial and not isomorphic to M^u_j. If neither M^u_j or M^u_k is trivial and the two are not isomorphic then the operation fails.

- $\texttt{Merge}(x, M_j^u, M_k^u)$ returns the map M_i^u with $i > j, i > k$ when M_j^u, M_k^u are consistent where if a is a node in M_j^u then $M_i^u(a) = M_j^u(a)$ and if a is a node in M_k^u then $M_i^u(a) = M_k^u(a)$. Merge map M_i^u has a new node $r(u, i)$ as a root node (which is greater than the maximum node in each of $M_i^u(a)$ or $M_j^u(a)$), and is defined as $M_i^u(r(u, i)) = (x, r(u, j), r(u, k))$.

Proofs in M-Res consist of lines, where every line is a pair $(C_i, \{M_i^u \mid u \in U\})$. Here, C_i is a purely existential clause (it contains only literals that are from existentially quantified variables). The other part is a set containing merge maps for each universal variable (some of the merge maps can be trivial, meaning they do not represent any function). Each line is derived by one of two rules:

Axiom: $C_i = \{l \mid l \in C, \text{var}(l) \in E\}$ is the existential subset of some clause C where C is a clause in the matrix. If universal literals $u, \bar{u}$ do not appear in C, let M_i^u be trivial. If universal variable u appears in C then let i be the sole node of M_i^u with $M_i^u(i) = \bot$. Likewise if $\neg u$ appears in C then let i be the sole node of M_i^u with $M_i^u(i) = \top$.

Resolution: Two lines $(C_j, \{M_j^u \mid u \in U\})$ and $(C_k, \{M_k^u \mid u \in U\})$ can be resolved to obtain a line $(C_i, \mid \{M_i^u \mid u \in U\})$ if there is literal $\neg x \in C_j$ and $x \in C_k$ such that $C_i = C_j \cup C_k \setminus \{x, \neg x\}$, and every M_i^u can either be defined as $\texttt{Select}(M_j^u, M_k^u)$, when M_j^u and M_k^u are isomorphic or one is trivial, or as $\texttt{Merge}(x, M_j^u, M_k^u)$ when $x < u$ and M_j^u and M_k^u are consistent.

3.2 Simulation of Merge Resolution

We now state the main result of this section.

▶ **Theorem 3.** eFrege + ∀red *simulates M-Res.*

For a false QBF $\Pi\phi$ refuted by M-Res, the final set of merge maps represent a falsifying strategy for the universal player, the strategy can be asserted by a proposition S that states that all universal variables are equivalent to their strategy circuits. It then should be the case that if ϕ is true, S must be false, a fact that can be proved propositionally, formally $\phi \vdash \neg S$.

To build up to this proof we can inductively find a local strategy S_i for each clause C_i that appears in an M-Res line $(C_i, \{M_i^u\})$ such that $\phi \vdash S_i \rightarrow C_i$. Elegantly, S_i is really just a circuit expressing that each $u \in U$ takes its value in M_i^u (if non-trivial). Extension variables are used to represent these local strategy circuits and so the proof ends up as a propositional extended Frege proof.

The final part of the proof is the technique suggested by Chew [12] which was originally used by Beyersdorff et al. [6]. That is, to use universal reduction starting from the negation of a universal strategy and arrive at the empty clause.

Proof.

Definition of extension variables. We create new extension variables for each node in every non-trivial merge map appearing in a proof. $s_{i,j}^u$ is created for the node j in merge map M_i^u. $s_{i,t}^u$ is defined as a constant when t is leaf node in M_i^u. Otherwise $s_{i,a}^u$ is defined as $s_{i,a}^u := (y \wedge s_{i,b}^u) \vee (\neg y \wedge s_{i,c}^u)$, when $M_i^u(j) = (y, b, c)$. Because y has to be before u in the prefix, $s_{i,j}^u$ is always defined before universal variable u.

Induction Hypothesis. It is easy for eFrege to prove $\bigwedge_{u \in U_i} (u \leftrightarrow s_{i,r(u,i)}^u) \rightarrow C_i$, where $r(u, i)$ is the index of the root node of Merge map M_i^u. U_i is the subset of U for which M_i^u is non-trivial.

Base Case: Axiom. Suppose C_i is derived by axiom download of clause C. If u has a strategy, it is because it appears in a clause and so $u \leftrightarrow s^u_{i,i}$, where $s^u_{i,i} \leftrightarrow c_u$ for $c_u \in \top, \bot$, c_u is correctly chosen to oppose the literal in C so that C_i is just the simplified clause of C replacing all universal u with their c_u. This is easy for eFrege to prove.

Inductive Step: Resolution. If C_j is resolved with C_k to get C_i with pivots $\neg x \in C_j$ and $x \in C_k$, we first show $\bigwedge_{u\in U_i}(u \leftrightarrow s^u_{i,r(u,i)}) \rightarrow C_j$ and $\bigwedge_{u\in U_i}(u \leftrightarrow s^u_{i,r(u,i)}) \rightarrow C_k$, where $r(u,i)$ is the root index of the Merge map for u on line i. We resolve these together.

To argue that $\bigwedge_{u\in U_i}(u \leftrightarrow s^u_{i,r(u,i)}) \rightarrow C_j$ we prove by induction that we can replace $u \leftrightarrow s^u_{j,r(u,j)}$ with $u \leftrightarrow s^u_{i,r(u,i)}$ one by one.

Induction Hypothesis. U_i is partitioned into W the set of adjusted variables and V the set of variables yet to be adjusted.

$(\bigwedge_{v\in V\cap U_j}(v \leftrightarrow s^v_{j,r(v,j)})) \wedge (\bigwedge_{v\in W}(v \leftrightarrow s^v_{i,r(v,i)})) \rightarrow C_j$

Base Case. $(\bigwedge_{v\in U_i\cap U_j}(v \leftrightarrow s^v_{j,r(v,j)}) \rightarrow C_j$ is the premise of the (outer) induction hypothesis, since $U_j \subseteq U_i$.

Inductive Step. Starting with $(\bigwedge_{v\in V\cap U_j}(v \leftrightarrow s^v_{j,r(v,j)})) \wedge (\bigwedge_{w\in W}(w \leftrightarrow s^w_{i,r(w,i)})) \rightarrow C_j$ We pick a $u \in V$ to show $(u \leftrightarrow s^w_{i,r(u,i)}) \wedge (\bigwedge^{v\neq u}_{v\in V\cap U_j}(v \leftrightarrow s^v_{j,r(v,j)})) \wedge (\bigwedge_{w\in W}(w \leftrightarrow s^w_{i,r(w,i)})) \rightarrow C_j$ We have four cases:

1. Select chooses $M^u_i = M^u_j$
2. Select chooses $M^u_i = M^u_k$ because M^u_j is trivial
3. Select chooses $M^u_i = M^u_k$ because there is an isomorphism f that maps M^u_j to M^u_k.
4. Merge so that M^u_i is the merge of M^u_j and M^u_k over pivot x

In (1) we prove inductively from the leaves to the root that $s^u_{i,t} \leftrightarrow s^u_{j,t}$. Eventually, we end up with $s^u_{i,r(u,i)} \leftrightarrow s^u_{j,r(u,j)}$. Then $(u \leftrightarrow s^u_{j,r(u,j)})$ can be replaced by $(u \leftrightarrow s^u_{i,r(u,i)})$.

In (2) we are simply weakening the implication as $(u \leftrightarrow s^u_{j,r(u,j)})$ never appeared before.

In (3) we prove inductively from the leaves to the root that $s^u_{i,f(t)} = s^u_{k,f(t)} = s^u_{j,t}$. Eventually, we end up with $s^u_{i,f(r(u,i))} = s^u_{k,f(r(u,i))} = s^u_{j,r(u,i)}$. Then $(u \leftrightarrow s^u_{j,r(u,j)})$ can be replaced by $(u \leftrightarrow s^u_{i,f(r(u,j))})$. As f is an isomorphism $f(r(u,j)) = r(u,k)$ and because Select is used $r(u,k) = r(u,i)$. Therefore we have $(u \leftrightarrow s^u_{i,r(u,i)})$.

In (4) we prove inductively that for each node t in M^u_j we have $(s^u_{i,t} \leftrightarrow s^u_{j,t})$. This is true in all leaf nodes as $s^u_{i,t}$ and $s^u_{j,t}$ have the same constant value. For intermediate nodes a, $s^u_{j,a} := (y \wedge s^u_{j,b}) \vee (\neg y \wedge s^u_{j,c})$ where b and c are other nodes. Since M^u_i is consistent with M^u_j then $s^u_{i,a} := (y \wedge s^u_{i,b}) \vee (\neg y \wedge s^u_{i,c})$ and since $s^u_{i,b} \leftrightarrow s^u_{j,b}$ and $s^u_{i,c} \leftrightarrow s^u_{j,c}$ by induction hypothesis, we have $s^u_{i,a} \leftrightarrow s^u_{j,a}$. eventually we have $s^u_{i,r(u,j)} \leftrightarrow s^u_{j,r(u,j)}$. However we need to replace $s^u_{j,r(u,j)}$ with $s^u_{i,r(u,i)}$, not $s^u_{i,r(u,j)}$. For this we use the definition of merging that $x \rightarrow (s^u_{i,r(u,i)} \leftrightarrow s^u_{i,r(u,j)})$ and so we have $(s^u_{i,r(u,i)} \leftrightarrow s^u_{j,r(u,j)}) \vee \neg x$ but the $\neg x$ is absorbed by the C_j in right hand side of the implication.

Finalise Inner Induction. At the end of this inner induction, we have $\bigwedge_{u\in U_i}(u \leftrightarrow s^u_{i,r(u,i)}) \rightarrow C_j$ and symmetrically $\bigwedge_{u\in U_i}(u \leftrightarrow s^u_{i,r(u,i)}) \rightarrow C_k$. We can then prove $\bigwedge_{u\in U_i}(u \leftrightarrow s^u_{i,r(u,i)}) \rightarrow C_i$.

Finalise Outer Induction. Note that we have done three nested inductions on the nodes in a merge maps, on the the universal variables, and then on the lines of an M-Res proof. Nonetheless, this gives a linear size eFrege proof in the number of nodes appearing in the proof. In M-Res the final line will be the empty clause and its merge maps. The induction gives us $\bigwedge_{u\in U_l}(u \leftrightarrow s^u_{l,r(u,l)}) \rightarrow \bot$. In other words, if $U_l = \{y_1, \dots y_n\}$, where y_i appears before y_{i+1} in the prefix, $\bigvee^n_{i=1}(y_i \oplus s^{y_i}_{l,r(y_i,l)})$.

We derive $(0\oplus s^{y_{n-k+1}}_{l,r(y_{n-k+1},l)})\vee\bigvee_{i=1}^{n-k}(y_i\oplus s^{y_i}_{l,r(y_i,l)})$ and $(1\oplus s^{y_{n-k+1}}_{l,r(y_{n-k+1},l)})\vee\bigvee_{i=1}^{n-k}(y_i\oplus s^{y_i}_{l,r(y_i,l)})$ from reduction of $\bigvee_{i=1}^{n-k+1}(y_i\oplus s^{y_i}_{l,r(y_i,l)})$. We can resolve both with the easily proved tautology $\bigvee_{i=1}^{n-k}(y_i \oplus s^{y_i}_{l,r(y_i,l)})$. We continue this until we reach the empty disjunction. ◀

4 Extended Frege+∀-Red p-simulates Expansion Based Systems

4.1 Expansion-Based Resolution Systems

The idea of an expansion based QBF proof system is to utilise the semantic identity: $\forall u\phi(u) = \phi(0) \wedge \phi(1)$, to replace universal quantifiers and their variables with propositional formulas. With $\forall u\exists x\phi(u) = \exists x\phi(0)\wedge\exists x\phi(1)$ the x from $\exists x\phi(0)$ and from $\exists x\phi(1)$ are actually different variables. The way to deal with this while maintaining prenex normal form is to introduce annotations that distinguish one x from another.

▶ **Definition 4.**

1. *An* extended assignment *is a partial mapping from the universal variables to* $\{0, 1, *\}$. *We denote an extended assignment by a set or list of individual replacements i.e.* $0/u, */v$ *is an extended assignment.*
2. *An* annotated clause *is a clause where each literal is annotated by an extended assignment to universal variables.*
3. *For an extended assignment* σ *to universal variables we write* $l^{\mathsf{restrict}_l(\sigma)}$ *to denote an annotated literal where* $\mathsf{restrict}_l(\sigma) = \{c/u \in \sigma \mid \mathrm{lv}(u) < \mathrm{lv}(l)\}$.
4. *Two (extended) assignments* τ *and* μ *are called* contradictory *if there exists a variable* $x \in \mathsf{dom}(\tau) \cap \mathsf{dom}(\mu)$ *with* $\tau(x) \neq \mu(x)$.

4.1.1 Definitions

The most simple way to use expansion would be to expand all universal quantifiers and list every annotated clause. The first expansion based system we consider, ∀Exp+Res, has a mechanism to avoid a this potential exponential explosion in some (but not all) cases. An annotated clause is created and then checked to see if it could be obtained from expansion. This way a refutation can just use an unsatisfiable core rather than all clauses from a fully expanded matrix.

$$\frac{}{\left\{l^{\mathsf{restrict}_l(\tau)} \mid l \in C, l \text{ is existential}\right\} \cup \{\tau(l) \mid l \in C, l \text{ is universal}\}}\ \text{(Axiom)}$$

C is a clause from the matrix and τ is an assignment to all universal variables.

$$\frac{C_1 \cup \{x^\tau\} \qquad C_2 \cup \{\neg x^\tau\}}{C_1 \cup C_2}\ \text{(Res)}$$

Figure 2 The rules of ∀Exp+Res (adapted from [21]).

The drawback of ∀Exp+Res is that one might end up repeating almost the same derivations over and over again if they vary only in changes in the annotation which make little difference in that part of the proof. This was used to find a lower bound to ∀Exp+Res for a family of formulas easy in system Q-Res [21]. To rectify this, IR-calc improved on ∀Exp+Res to allow a delay to the annotations in certain circumstances. Annotated clauses now have annotations with "gaps" where the value of the universal variable is yet to be set. When they are set there is the possibility of choosing both assignments without the need to rederive the annotated clauses with different annotations.

▶ **Definition 5.** *Given two partial assignments (or partial annotations) α and β. The completion $\alpha \circ \beta$, is a new partial assignment, where*

$$\alpha \circ \beta(u) = \begin{cases} \alpha(u) & \text{if } u \in \mathsf{dom}(\alpha) \\ \beta(u) & \text{if } u \in \mathsf{dom}(\beta) \setminus \mathsf{dom}(\alpha) \\ \text{unassigned} & \text{otherwise} \end{cases}$$

For α an assignment of the universal variables and C an annotated clause we define $\mathsf{inst}(\alpha, C) := \bigvee_{l^\tau \in C} l^{\mathsf{restrict}_l(\tau \circ \alpha)}$. Annotation α here gives values to unset annotations where one is not already defined. Because the same α is used throughout the clause, the previously unset values gain consistent annotations, but mixed annotations can occur due to already existing annotations.

$$\frac{}{\{l^{\mathsf{restrict}_l(\tau)} \mid l \in C, l \text{ is existential}\}} \text{(Axiom)}$$

C is a non-tautological clause from the matrix. $\tau = \{0/u \mid u \text{ is universal in } C\}$, where the notation $0/u$ for literals u is shorthand for $0/x$ if $u = x$ and $1/x$ if $u = \neg x$.

$$\frac{x^\tau \vee C_1 \qquad \neg x^\tau \vee C_2}{C_1 \cup C_2} \text{(Resolution)} \qquad \frac{C}{\mathsf{inst}(\tau, C)} \text{(Instantiation)}$$

τ is an assignment to universal variables with $\mathsf{rng}(\tau) \subseteq \{0, 1\}$.

Figure 3 The rules of IR-calc [7].

The definition of IR-calc is given in Figure 3. Resolved variables have to match exactly, including that missing values are missing in both pivots. However, non-contradictory but different annotations may still be used for a later resolution step after the instantiation rule is used to make the annotations match the annotations of the pivot.

4.1.2 Local Strategies and Policies

The work from Schlaipfer et al. [34] creates a conversion of each annotated clause C into a propositional formula con(C) defined in the original variables of ϕ (so without creating new annotated variables). C appearing in a proof asserts that there is some (not necessarily winning) strategy for the universal player to force con(C) to be true under ϕ. The idea is that for each line C in an ∀Exp+Res refutation of $\Pi\phi$ there is some local strategy S such that $S \wedge \phi \rightarrow \text{con}(C)$.

The construction of the strategy is formed from the structure of the proof and follows the semantic ideas of Suda and Gleiss [37], in particular the `Combine` operation for resolution. The extra work by Schlaipfer et al. is that the strategy circuits (for each u) can be constructed in polynomial time, and can be defined in variables left of u_i in the prefix.

Let $u_1 \dots u_n$ be all universal variables in order. For each line in an ∀Exp+Res proof we have a strategy which we will here call S. For each u_i there is an extension variable Val^i_S, before u_i, that represents the value assigned to u_i by S (under an assignment of existential variables). Using these variables, we obtain a propositional formula representing the strategy as $S = \bigwedge_{i=1}^{n} u_i \leftrightarrow \mathrm{Val}_S^i$. Additionally, we define a conversion of annotated logic in $\forall$Exp+Res to propositional logic as follows. For annotations τ let $\mathrm{anno}(\tau) = \bigwedge_{1/u_i \in \tau} u_i \wedge \bigwedge_{0/u_i \in \tau} \bar{u}_i$. We convert annotated literals as $\mathrm{con}(l^\tau) = l \wedge \mathrm{anno}(\tau)$ and clauses as $\mathrm{con}(C) = \bigvee_{l \in C} \mathrm{con}(l)$.

4.2 Simulating IR-calc

The conversion needs to be revised for IR-calc. In particular the variables not set in the annotations need to be understood. The solution is to basically treat unset as a third value, although in practice this requires new Set_S^i variables (left of u_i) which state that the ith universal variable is set by policy S. We include these variables in our encoding of policy S and let $S = \bigwedge_{i=1}^{n} \mathrm{Set}_S^i \rightarrow (u_i \leftrightarrow \mathrm{Val}_S^i)$. The conversion of annotations, literals and clauses also has to be changed. For annotations τ let

$$\mathrm{anno}_{x,S}(\tau) = \bigwedge\nolimits_{1/u_i \in \tau} (\mathrm{Set}_S^i \wedge u_i) \wedge \bigwedge\nolimits_{0/u_i \in \tau} (\mathrm{Set}_S^i \wedge \bar{u}_i) \wedge \bigwedge\nolimits_{u_i <_\Pi x}^{u_i \notin \mathsf{dom}(\tau)} \neg \mathrm{Set}_S^i .$$

Let $\mathrm{con}_S(l^\tau) = l \wedge \mathrm{anno}_{x,S}(\tau)$ and $\mathrm{con}_S(C) = \bigvee_{l \in C} \mathrm{con}_S(l)$ similarly to before, we just reference a particular policy S. This means that we again want $S \wedge \phi \rightarrow \mathrm{con}_S(C)$ for each line, note that Set_S^i variables are defined in their own way.

The most crucial part of simulating IR-calc is that after each application of the resolution rule we can obtain a working policy.

▶ **Lemma 6.** *Suppose, there are policies L and R such that $L \rightarrow \mathrm{con}_L(C_1 \vee \neg x^\tau)$ and $R \rightarrow \mathrm{con}_R(C_1 \vee x^\tau)$ then there is a policy B such that $B \rightarrow \mathrm{con}_B(C_1 \vee C_2)$ can be obtained in a short* eFrege *proof.*

The proof of the simulation of IR-calc relies on Lemma 6. To prove this we have to first give the precise definitions of the policy B based on policies L and R. Schlaipfer et al.'s work [34] is used to crucially make sure the strategy B, respects the prefix ordering.

4.2.1 Building the Strategy

We start to define Val_B^i and Set_B^i on lower i values first. In particular we will always start with $1 \leq i \leq m$ where u_m is the rightmost universal variable still before x in the prefix. Starting from $i = 0$, the initial segments of $\mathrm{con}_{x,L}(\tau)$ and $\mathrm{con}_{x,R}(\tau)$ may eventually reach such a point j where one is contradicted. Before this point L and R are detailing the same strategy (they may differ on Val^i but only when Set^i is false) so B can be played as both simultaneously as L and R. Without loss of generality, as soon as L contradicts $\mathrm{anno}_{x,L}(\tau)$, we know that $\mathrm{con}_L(x^\tau)$ is not satisfied by L and thus it makes sense for B to copy L, at this point and the rest of the strategy as it will satisfy $\mathrm{con}_B(C_1)$. It is entirely possible that we reach $i = m$ and not contradict either $\mathrm{con}_{x,L}(\tau)$ or $\mathrm{con}_{x,R}(\tau)$. Fortunately after this point in the game we now know the value the existential player has chosen for x. We can use the x value to decide whether to play B as L (if x is true) or R (if x is false).

To build the circuitry for Val_B^i and Set_B^i we will introduce other circuits that will act as intermediate. First we will use constants Set_τ^i and Val_τ^i that make $\mathrm{anno}_{x,S}(\tau)$ equivalent to $\bigwedge_{u_i <_\Pi x} (\mathrm{Set}_S^i \leftrightarrow \mathrm{Set}_\tau^i) \wedge \mathrm{Set}_\tau^i \rightarrow (u_i \leftrightarrow \mathrm{Val}_\tau^i)$. This mainly makes our notation easier. Next we will define circuits that represent two strategies being equivalent up to the ith universal variable. This is a generalisation of what was seen in the local strategy extraction for $\forall$Exp+Res [34].
$\mathrm{Eq}_{f=g}^0 := 1$, $\mathrm{Eq}_{f=g}^i := \mathrm{Eq}_{f=g}^{i-1} \wedge (\mathrm{Set}_f^i \leftrightarrow \mathrm{Set}_g^i) \wedge (\mathrm{Set}_f^i \rightarrow (\mathrm{Val}_f^i \leftrightarrow \mathrm{Val}_g^i))$.

We specifically use this for a trigger variable that tells you which one of L and R differed from τ first.
$\mathrm{Dif}^0_L := 0$ and $\mathrm{Dif}^i_L := \mathrm{Dif}^{i-1}_L \vee (\mathrm{Eq}^{i-1}_{R=\tau} \wedge ((\mathrm{Set}^i_L \oplus \mathrm{Set}^i_\tau) \vee (\mathrm{Set}^i_\tau \wedge (\mathrm{Val}^i_L \oplus \mathrm{Val}^i_\tau))))$
$\mathrm{Dif}^0_R := 0$ and $\mathrm{Dif}^i_R := \mathrm{Dif}^{i-1}_R \vee (\mathrm{Eq}^{i-1}_{L=\tau} \wedge ((\mathrm{Set}^i_R \oplus \mathrm{Set}^i_\tau) \vee (\mathrm{Set}^i_\tau \wedge (\mathrm{Val}^i_R \oplus \mathrm{Val}^i_\tau))))$

Note that Dif^i_L and Dif^i_R can both be true but only if they start to differ at the same point.

Suda and Gleiss's `Combine` operation allows one to construct a bottom policy B that chooses between the left and right policies.

▶ **Definition 7** (Definition of resolvent policy for IR-calc). *For* $0 \leq i \leq m$, *define* Val^i_B *and* Set^i_B *such* $\mathrm{Val}^i_B = \mathrm{Val}^i_R$ *and* $\mathrm{Set}^i_B = \mathrm{Set}^i_R$ *if*

$$\neg \underset{L}{\overset{i-1}{\mathrm{Dif}}} \wedge (\underset{R}{\overset{i-1}{\mathrm{Dif}}} \vee (\neg \underset{\tau}{\overset{i}{\mathrm{Set}}} \wedge \neg \underset{L}{\overset{i}{\mathrm{Set}}} \wedge \underset{R}{\overset{i}{\mathrm{Set}}}) \vee (\underset{\tau}{\overset{i}{\mathrm{Set}}} \wedge \underset{L}{\overset{i}{\mathrm{Set}}} \wedge (\underset{\tau}{\overset{i}{\mathrm{Val}}} \leftrightarrow \underset{L}{\overset{i}{\mathrm{Val}}})))$$

and $\mathrm{Val}^i_B = \mathrm{Val}^i_L$ *and* $\mathrm{Set}^i_B = \mathrm{Set}^i_L$, *otherwise.*
For $i > m$, *define* Val^i_B *and* Set^i_B *such* $\mathrm{Val}^i_B = \mathrm{Val}^i_R$ *and* $\mathrm{Set}^i_B = \mathrm{Set}^i_R$ *if*

$$\neg \underset{L}{\overset{m}{\mathrm{Dif}}} \wedge (\underset{R}{\overset{m}{\mathrm{Dif}}} \vee \bar{x})$$

and $\mathrm{Val}^i_B = \mathrm{Val}^i_L$ *and* $\mathrm{Set}^i_B = \mathrm{Set}^i_L$, *otherwise.*

We will now define variables B_L and B_R. These say that B is choosing L or R, respectively. These variables can appear rightmost in the prefix, as they will be removed before reduction takes place. The purpose of B_L (resp. B_R) is that con_B becomes the same as con_L (resp. con_R).

- $B_L := \bigwedge_{i=1}^n (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_L) \wedge (\mathrm{Set}^i_B \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_L))$
- $B_R := \bigwedge_{i=1}^n (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_R) \wedge (\mathrm{Set}^i_B \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_R))$

We have not fully defined B here. The important points are that B is set up so that it either takes values in L or R , i.e. $B \rightarrow B_L \vee B_R$, specifically we need that whenever the propositional formula $\mathrm{anno}_{x,B}(\tau)$ is satisfied, $B = B_L$ when x, and $B = B_R$ when $\neg x$. The variables Set^i_B and Val^i_B that comprise the policy are carefully constructed to come before u_i. A number of technical lemmas involving all these definitions is necessary for the simulation.

▶ **Lemma 8.** *For* $0 < j \leq m$ *the following propositions have short derivations in Extended Frege:*

- $\mathrm{Dif}^j_L \rightarrow \bigvee_{i=1}^j \mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L$
- $\mathrm{Dif}^j_R \rightarrow \bigvee_{i=1}^j \mathrm{Dif}^i_R \wedge \neg \mathrm{Dif}^{i-1}_R$
- $\neg \mathrm{Eq}^j_{f=g} \rightarrow \bigvee_{i=1}^j \neg \mathrm{Eq}^i_{f=g} \wedge \mathrm{Eq}^{i-1}_{f=g}$. *For* $f, g \in \{L, R, \tau\}$.

▶ **Lemma 9.** *For* $0 \leq i \leq j \leq m$ *the following propositions that describe the monotonicity of* Dif *have short derivations in Extended Frege:*

- $\mathrm{Dif}^i_L \rightarrow \mathrm{Dif}^j_L$
- $\mathrm{Dif}^i_R \rightarrow \mathrm{Dif}^j_R$
- $\neg \mathrm{Eq}^i_{f=g} \rightarrow \neg \mathrm{Eq}^j_{f=g}$

▶ **Lemma 10.** *For* $0 \leq i \leq j \leq m$ *the following propositions describe the relationships between the different extension variables and have short derivations in Extended Frege:*

- $\mathrm{Eq}^i_{L=\tau} \rightarrow \neg \mathrm{Dif}^i_L$
- $\mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L \rightarrow \mathrm{Eq}^{i-1}_{R=\tau}$
- $\mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L \rightarrow \neg \mathrm{Dif}^{i-1}_R$

- $\mathrm{Eq}^i_{R=\tau} \rightarrow \neg\,\mathrm{Dif}^i_R$
- $\mathrm{Dif}^i_R \wedge \neg\,\mathrm{Dif}^{i-1}_R \rightarrow \mathrm{Eq}^{i-1}_{L=\tau}$
- $\mathrm{Dif}^i_R \wedge \neg\,\mathrm{Dif}^{i-1}_R \rightarrow \neg\,\mathrm{Dif}^{i-1}_L$

▶ **Lemma 11.** *For any $0 \leq i \leq m$ the following propositions are true and have short Extended Frege proofs.*

- $L \wedge \mathrm{Dif}^i_L \rightarrow \neg\,\mathrm{anno}_{x,L}(\tau)$
- $R \wedge \mathrm{Dif}^i_R \rightarrow \neg\,\mathrm{anno}_{x,R}(\tau)$

▶ **Lemma 12.** *For any $1 \leq j \leq m$ the following propositions are true and have a short Extended Frege proof.*

- $\neg\,\mathrm{Dif}^j_L \wedge \neg\,\mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_L$
- $\neg\,\mathrm{Dif}^j_L \wedge \neg\,\mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_R$
- $\neg\,\mathrm{Dif}^j_L \wedge \neg\,\mathrm{Dif}^j_R \rightarrow (\mathrm{Set}^j_B \leftrightarrow \mathrm{Set}^j_L)$
- $\neg\,\mathrm{Dif}^j_L \wedge \neg\,\mathrm{Dif}^j_R \rightarrow \mathrm{Set}^i_B \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_L)$
- $\neg\,\mathrm{Dif}^j_L \wedge \neg\,\mathrm{Dif}^j_R \rightarrow (\mathrm{Set}^j_B \leftrightarrow \mathrm{Set}^j_R)$
- $\neg\,\mathrm{Dif}^j_L \wedge \neg\,\mathrm{Dif}^j_R \rightarrow \mathrm{Set}^i_B \rightarrow (\mathrm{Val}^j_B \leftrightarrow \mathrm{Val}^j_R)$

▶ **Lemma 13.** *For any $0 \leq i \leq m$ the following propositions are true and have short Extended Frege proofs.*

- $\mathrm{Dif}^i_L \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_L) \wedge (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_L)$
- $\neg\,\mathrm{Dif}^i_L \wedge \mathrm{Dif}^i_R \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_R) \wedge (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_R)$

▶ **Lemma 14.** *The following propositions are true and have short Extended Frege proofs.*

- $B \wedge \mathrm{Dif}^m_L \rightarrow B_L$
- $B \wedge \neg\,\mathrm{Dif}^m_L \wedge \mathrm{Dif}^m_R \rightarrow B_R$

▶ **Lemma 15.** *The following propositions are true and have short Extended Frege proofs.*

- $B \wedge \neg\,\mathrm{Dif}^m_L \wedge \neg\,\mathrm{Dif}^m_R \rightarrow B_L \vee \neg x$
- $B \wedge \neg\,\mathrm{Dif}^m_L \wedge \neg\,\mathrm{Dif}^m_R \rightarrow B_R \vee x$

▶ **Lemma 16.** *The following proposition is true and has a short Extended Frege proof.* $B \rightarrow B_L \vee B_R$

Proof. This roughly says that B either is played entirely as L or is played as R. We can prove this by combining Lemmas 14 and 15, it essentially is a case analysis in formal form. ◀

▶ **Lemma 17.** *The following propositions are true and have short Extended Frege proofs.*

- $B \wedge \mathrm{anno}(\tau) \wedge x \rightarrow B_L,$
- $B \wedge \mathrm{anno}(\tau) \wedge \neg x \rightarrow B_R$

Proof. We start with $B \wedge \neg\,\mathrm{Dif}^m_L \wedge \neg\,\mathrm{Dif}^m_R \rightarrow B_L \vee \neg x$ and $B \wedge \neg\,\mathrm{Dif}^m_L \wedge \neg\,\mathrm{Dif}^m_R \rightarrow B_R \vee x$. It remains to remove $\neg\,\mathrm{Dif}^m_L \wedge \neg\,\mathrm{Dif}^m_R$ from the left hand side. This is where we use $L \wedge \mathrm{Dif}^i_L \rightarrow \neg\,\mathrm{anno}_L(\tau)$ and $R \wedge \mathrm{Dif}^i_R \rightarrow \neg\,\mathrm{anno}_R(\tau)$ from Lemma 11. These can be simplified to $B \wedge B_L \wedge \mathrm{Dif}^m_L \rightarrow \neg\,\mathrm{anno}_B(\tau)$ and $B \wedge B_R \wedge \mathrm{Dif}^m_R \rightarrow \neg\,\mathrm{anno}_B(\tau)$. The B_L and B_R can be removed by using $B \wedge \mathrm{Dif}^m_L \rightarrow B_L$ and $B \wedge \neg\,\mathrm{Dif}^m_L \wedge \mathrm{Dif}^m_R \rightarrow B_R$ and we can end up with $B \rightarrow \neg\,\mathrm{anno}_B(\tau) \vee (\neg\,\mathrm{Dif}^m_R \wedge \neg\,\mathrm{Dif}^m_L)$ we can use this to resolve out $(\neg\,\mathrm{Dif}^m_R \wedge \neg\,\mathrm{Dif}^m_L)$ and get $B \wedge \mathrm{anno}(\tau) \wedge x \rightarrow B_L$ and $B \wedge \mathrm{anno}(\tau) \wedge \neg x \rightarrow B_R$. ◀

Proof of Lemma 6. Since $B \wedge B_L \rightarrow L$ and $B \wedge B_R \rightarrow R$, $L \rightarrow \mathrm{con}_L(C_1 \vee \neg x^\tau)$ and $R \rightarrow \mathrm{con}_L(C_2 \vee x^\tau)$ imply $B \wedge B_L \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \mathrm{anno}_{x,B}(\tau)$, $B \wedge B_R \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \mathrm{anno}_{x,B}(\tau)$, $B \wedge B_L \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \neg x$ and $B \wedge B_R \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee x$.

We combine $B \rightarrow B_L \vee B_R$ with $B \wedge B_L \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \mathrm{anno}_{x,B}(\tau)$ (removing B_L) and $B \wedge B_R \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \mathrm{anno}_{x,B}(\tau)$ (removing B_R) to gain $B \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \mathrm{anno}_{x,B}(\tau)$. Next, we derive $B \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \neg\, \mathrm{anno}_{x,B}(\tau)$. Policy B is set up so that $B \wedge \mathrm{anno}_{x,B}(\tau) \wedge x \rightarrow B_L$ and $B \wedge \mathrm{anno}_{x,B}(\tau) \wedge \neg x \rightarrow B_R$ have short proofs. We resolve these, respectively, with $B \wedge B_R \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee x$ (on x) to obtain $B \wedge \mathrm{anno}_{x,B}(\tau) \wedge B_R \rightarrow B_L \vee \mathrm{con}_B(C_1 \vee C_2)$, and with $B \wedge B_L \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \neg x$ (on $\neg x$) to obtain $B \wedge \mathrm{anno}_{x,B}(\tau) \wedge B_L \rightarrow B_R \vee \mathrm{con}_B(C_1 \vee C_2)$. Putting these together allows us to remove B_L and B_R, deriving $B \wedge \mathrm{anno}_{x,B}(\tau) \rightarrow \mathrm{con}_B(C_1 \vee C_2)$, which can be rewritten as $B \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \neg\, \mathrm{anno}_{x,B}(\tau)$.

We now have two formulas $B \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \neg\, \mathrm{anno}_{x,B}(\tau)$ and $B \rightarrow \mathrm{con}_B(C_1 \vee C_2) \vee \mathrm{anno}_{x,B}(\tau)$, which resolve to get $B \rightarrow \mathrm{con}_B(C_1 \vee C_2)$. ◀

▶ **Theorem 18.** eFrege + ∀red *p-simulates IR-calc.*

Proof. We prove by induction that every annotated clause C appearing in an IR-calc proof has a local policy S such that $\phi \vdash_{\mathsf{eFrege}} S \rightarrow \mathrm{con}_S(C)$ and this can be done in a polynomial-size proof.

Axiom. Suppose $C \in \phi$ and $D = \mathsf{inst}(C, \tau)$ for partial annotation τ. We construct policy B such that $B \rightarrow \mathrm{con}_B(D)$.

$$\underset{B}{\overset{j}{\mathrm{Set}}} = \begin{cases} 1 & \text{if } u_j \in \mathsf{dom}(\tau) \\ 0 & u_j \notin \mathsf{dom}(\tau) \end{cases}, \ \mathrm{Val}^j_B = \begin{cases} 1 & \text{if } 1/u_j \in \tau \\ 0 & \text{if } 0/u_j \in \tau \end{cases}$$

Instantiation. Suppose we have an instantiation step for C on a single universal variable u_i using instantiation $0/u_i$, so the new annotated clause is $D = \mathsf{inst}(C, 0/u_i)$. From the induction hypothesis $T \rightarrow \mathrm{con}_T(C)$ we will develop B such that $B \rightarrow \mathrm{con}_B(D)$.

$$\underset{B}{\overset{j}{\mathrm{Set}}} = \begin{cases} 1 & \text{if } j = i \\ \mathrm{Set}^j_T & \text{if } j \neq i \end{cases}, \ \mathrm{Val}^j_B = \begin{cases} \mathrm{Val}^j_T \wedge \mathrm{Set}^j_T & \text{if } j = i \\ \mathrm{Val}^j_T & \text{if } j \neq i \end{cases}$$

$\mathrm{Val}^j_T \wedge \mathrm{Set}^j_T$ becomes $\mathrm{Val}^j_T \vee \neg\, \mathrm{Set}^j_T$ for instantiation by $1/u_j$. Either case means B satisfies the same annotations anno as T appearing in our converted clauses $\mathrm{con}_B(C)$ and $\mathrm{con}_B(D)$, proving the rule as an inductive step.

Resolution. See Lemma 6.

Contradiction. At the end of the proof we have $T \rightarrow \mathrm{con}_T(\bot)$. T is a policy, so we turn it into a full strategy B by having for each i: $\mathrm{Val}^i_B \leftrightarrow (\mathrm{Val}^i_T \wedge \mathrm{Set}^i_T)$ and $\mathrm{Set}^i_B = 1$. Effectively this instantiates $\bot$ by the assignment that sets everything to 0 and we can argue that $B \rightarrow \mathrm{con}_B(\bot)$ although $\mathrm{con}_B(\bot)$ is just the empty clause. So we have $\neg B$. But $\neg B$ is just $\bigvee_{i=1}^n (u_i \oplus \mathrm{Val}^i_B)$. Furthermore, just as in Schlaipfer et al.'s work , we have been careful with the definitions of the extension variables Val^i_B so that they are left of u_i in the prefix. In eFrege + ∀red we can use the reduction rule (this is the first time we use the reduction rule). We show an inductive proof of $\bigvee_{i=1}^{n-k} (u_i \oplus \mathrm{Val}^i_B)$ for increasing k eventually leaving us with the empty clause. This essentially is where we use the ∀-Red rule. Since we already have $\bigvee_{i=1}^n (u_i \oplus \mathrm{Val}^i_B)$ we have the base case and we only need to show the inductive step.

We derive from $\bigvee_{i=1}^{n+1-k} (u_i \oplus \mathrm{Val}^i_B)$ both $(0 \oplus \mathrm{Val}^{n-k+1}_B) \vee \bigvee_{i=1}^{n-k} (u_i \oplus \mathrm{Val}^i_B)$ and $(1 \oplus \mathrm{Val}^{n-k+1}_B) \vee \bigvee_{i=1}^{n-k} (u_i \oplus \mathrm{Val}^i_B)$ from reduction. We can resolve both with the easily proved tautology $(0 \leftrightarrow \mathrm{Val}^{n-k+1}_B) \vee (1 \leftrightarrow \mathrm{Val}^{n-k+1}_B)$ which allows us to derive $\bigvee_{i=1}^{n-k} (u_i \oplus \mathrm{Val}^i_B)$.

We continue this until we reach the empty disjunction. ◀

▶ **Corollary 19.** eFrege + ∀red *p-simulates* ∀*Exp+Res.*

While this can be proven as a corollary of the simulation of IR-calc, a more direct simulation can be achieved by defining the resolvent strategy by removing the Set^i variables (i.e. by considering them as always true).

4.3 Simulating IRM-calc

4.3.1 Definition

IRM-calc was designed to compress annotated literals in clauses in order simulate LD-Q-Res. Like that system it uses the $*$ symbol, but since universal literals do not appear in an annotated clause, the $*$ value is added to the annotations, $0/u, 1/u, */u$ being the first three possibilities in an extended annotation (the fourth being when u does not appear in the annotation).

Axiom and instantiation rules as in IR-calc in Figure 3.

$$\frac{x^{\tau\cup\xi} \vee C_1 \qquad \neg x^{\tau\cup\sigma} \vee C_2}{\mathsf{inst}(\sigma, C_1) \cup \mathsf{inst}(\xi, C_2)} \text{ (Resolution)}$$

$\mathsf{dom}(\tau)$, $\mathsf{dom}(\xi)$ and $\mathsf{dom}(\sigma)$ are mutually disjoint. $\mathsf{rng}(\tau) = \{0, 1\}$

$$\frac{C \vee b^{\mu} \vee b^{\sigma}}{C \vee b^{\xi}} \text{ (Merging)}$$

$\mathsf{dom}(\mu) = \mathsf{dom}(\sigma)$. $\xi = \{c/u \mid c/u \in \mu, c/u \in \sigma\} \cup \{*/u \mid c/u \in \mu, d/u \in \sigma, c \neq d\}$

Figure 4 The rules of IRM-calc.

The rules of IRM-calc as given in Figure 4, become more complicated as a result of the $*/u$. In particular resolution is no longer done between matching pivots but matching is done internally in the resolution steps. This is to prevent variables resolving with matching $*$ annotations. Allowing such resolution steps would be unsound in general, as these $*$ annotations show that the universal variables are set according to some function, but when appearing in two different literals the functions could be completely different. Resolutions where one pivot literal has a $*/u$ annotation means that the other pivot literal must not have u in its annotation's domain. The intuition is that the unset u is given a $*$ value during the resolution but it can be controlled to be exactly the same $*$ as in the other pivot. A $0/u, 1/u$ or $*/u$ value cannot be given a new $*$ value so cannot match the other $*/u$ annotation.

It is in IRM-calc where the positive Set literals introduced in the simulation of IR-calc become useful. In most ways Set^i_S asserts the same things as $*/u_i$, that u_i is given a value, but this value does not have to be specified.

4.3.2 Conversion, Policies and Simulation

The first major change from IR-calc is that while con_S worked on three values in IR-calc, in IRM-calc we effectively run in four values $\mathrm{Set}^i_S, \neg\,\mathrm{Set}^i_S, \mathrm{Set}^i_S \wedge u_i$ and $\mathrm{Set}^i_S \wedge \neg u_i$. Set^i_S is the new addition deliberately ambiguous as to whether u_i is true or false. Readers familiar with the $*$ used in IRM-calc may notice why Set^i_S works as a conversion of $*/u_i$, as Set^i_S is just saying our policy has given a value but it may be different values in different circumstances.

Like in the case of IR-calc, most work needs to be done in the IRM-calc resolution steps, although here it is even more complicated. A resolution step in IRM-calc is in two parts. Firstly $C_1 \vee \neg x^{\tau \sqcup \sigma}$, $C_2 \vee x^{\tau \sqcup \xi}$ are both instantiated (but by $*$ in some cases), secondly they are resolved on a matching pivot. We simplify the resolution steps so that σ and ξ only contain $*$ annotations, for the other constant annotations that would normally be found in these steps suppose we have already instantiated them in the other side so that they now appear in τ (this does not affect the resolvent).

Again we assume that there are policies L and R such that $L \rightarrow \mathrm{con}_L(C_1 \vee \neg x^{\tau \sqcup \sigma})$ and $R \rightarrow \mathrm{con}_R(C_2 \vee x^{\tau \sqcup \xi})$. We know that if L falsifies $\mathrm{anno}_{x,L}(\tau \sqcup \sigma)$ then $\mathrm{con}_L(C_1)$ and likewise if R falsifies $\mathrm{anno}_{x,R}(\tau \sqcup \xi)$ then $\mathrm{con}_R(C_2)$ is satisfied. These are the safest options, however this leaves cases when L satisfies $\mathrm{anno}_{x,L}(\tau \sqcup \sigma)$ and R satisfies $\mathrm{anno}_{x,R}(\tau \sqcup \xi)$ but L and R are not equal. This happens either when Set^i_L and $\neg \mathrm{Set}^i_R$ both occur for $*/u_i \in \sigma$ or when $\neg \mathrm{Set}^i_L$ and Set^i_R both occur for $*/u_i \in \xi$.

This would cause issues if B had to choose between L and R to satisfy $\mathrm{con}_B(C_1 \vee C_2)$. Fortunately, we are not trying to satisfy $\mathrm{con}_B(C_1 \vee C_2)$ but $\mathrm{con}_B(\mathsf{inst}(\xi, C_1) \vee \mathsf{inst}(\sigma, C_2))$, so we have to choose between a policy that will satisfy $\mathrm{con}_B(\mathsf{inst}(\xi, C_1))$ and a policy that will satisfy $\mathrm{con}_B(\mathsf{inst}(\sigma, C_2))$. By borrowing values from the opposite policy we obtain a working new policy that does not have to choose between left and right any earlier than we would have for IR-calc.

▶ **Theorem 20.** eFrege + ∀red *simulates IRM-calc.*

▶ **Corollary 21.** eFrege + ∀red *simulates LD-Q-Res.*

5 Extended Frege+∀-Red p-simulates LQU$^+$-Res

5.1 QCDCL Resolution Systems

The most basic and important CDCL system is *Q-resolution (Q-Res)* by Kleine Büning et al. [25]. *Long-distance resolution (LD-Q-Res)* appears originally in the work of Zhang and Malik [40] and was formalized into a calculus by Balabanov and Jiang [2]. It merges complementary literals of a universal variable u into the special literal u^*. These special literals prohibit certain resolution steps. *QU-resolution (QU-Res)* [38] removes the restriction from Q-Res that the resolved variable must be an existential variable and allows resolution of universal variables. *LQU$^+$-Res* [3] extends LD-Q-Res by allowing short and long distance resolution pivots to be universal, however, the pivot is never a merged literal z^*. LQU$^+$-Res encapsulates Q-Res, LD-Q-Res and QU-Res.

5.2 Conversion to Propositional Logic and Simulation

LQU$^+$-Res and IRM-calc are mutually incomparable in terms of proof strength, however both share enough similarities to get the simulation working. Once again we can use Set^i variables to represent an u^*_i, and a $\neg \mathrm{Set}^i_S$ to represent that policy S chooses not to issue a value to u_i.

For any set of universal variables U, let $\mathrm{anno}_{x,S}(U) = \bigwedge_{u_j < x}^{u_j \notin U} \neg \mathrm{Set}^j_S \wedge \bigwedge_{u_j < x}^{u_j \in U} \mathrm{Set}^j_S$. Note that we do not really need to add polarities to the annotations, these are taken into account by the clause literals. Literals u and $\bar{u}$ do not need to be assigned by the policy, they are now treated as a consequence of the the CNF. Because they can be resolved we treat them like existential variables in the conversion. For universal variable u_i, $\mathrm{con}_{S,C}(u_i) = u_i \wedge \neg \mathrm{Set}^i_S \wedge \mathrm{anno}_{u_i,S}(\{u \mid u^* \in C\})$ and $\mathrm{con}_{S,C}(\neg u_i) = \neg u_i \wedge \neg \mathrm{Set}^i_S \wedge \mathrm{anno}_{u_i,S}(\{v \mid v^* \in C\})$. We reserve Set^j_S for starred literals as they cannot be removed. For existential literal x,

$\mathrm{con}_{S,C}(x) = x \wedge \mathrm{anno}_{x,S}(\{u \mid u^* \in C\})$. Finally, $\mathrm{con}_{S,C}(u^*) = \bot$, because we do not treat u^* as a literal but part of the "annotation" to literals right of it. Also, u^* cannot be resolved but it automatically reduced when no more literals are to the right of it. For clauses in LQU$^+$-Res, we let $\mathrm{con}_S(C) = \bigvee_{l \in C} \mathrm{con}_{S,C}(l)$. In summary, in comparison to IRM-calc the conversion now includes universal variables and gives them annotations, but removes polarities from the annotations. Policies still remain structured as they were for IR-calc, with extension variables Val^i_S and Set^i_S, where $S = \bigwedge_{i=1}^n \mathrm{Set}^i_S \to (u_i \leftrightarrow \mathrm{Val}^i_S)$.

▶ **Theorem 22.** eFrege + ∀red *simulates LQU$^+$-Res.*

6 Conclusion

Our work reconciles many different QBF proof techniques under the single system eFrege + ∀red. Although eFrege + ∀red itself is likely not a good system for efficient proof checking, our results have implications for other systems that are more promising in this regard, such as QRAT, which inherits these simulations. In particular, QRAT's simulation of ∀Exp+Res is upgraded to a simulation of IRM-calc, and we do not even require the extended universal reduction rule. Existing QRAT checkers can be used to verify converted eFrege + ∀red proofs. Further, extended QU-resolution is polynomially equivalent to eFrege + ∀red [12], and has previously been proposed as a system for unified QBF proof checking [22]. Since our simulations split off propositional inference from a standardised reduction part at the end, another option is to use (highly efficient) propositional proof checkers instead. Our simulations use many extension variables that are known to negatively impact the checking time of existing tools such as DRAT-trim, but one may hope that they can be refined to become more efficient in this regard.

There are other proof systems, particularly ones using dependency schemes, such as Q($\mathcal{D}^{\mathsf{rrs}}$)-Res and LD-Q($\mathcal{D}^{\mathsf{rrs}}$)-Res that have strategy extraction [31]. Local strategy extraction and ultimately a simulation by eFrege + ∀red seem likely for these systems, whether it can be proved directly or by generalising the simulation results from this paper.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

2 Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, 2012. doi:10.1007/s10703-012-0152-6.

3 Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *SAT 2014*, pages 154–169, 2014.

4 Olaf Beyersdorff. On the correspondence between arithmetic theories and propositional proof systems – a survey. *Mathematical Logic Quarterly*, 55(2):116–137, 2009.

5 Olaf Beyersdorff, Joshua Blinkhorn, and M. Mahajan. Building strategies into QBF proofs. In *Electron. Colloquium Comput. Complex.*, 2018.

6 Olaf Beyersdorff, Ilario Bonacina, Leroy Chew, and Jan Pich. Frege systems for quantified Boolean logic. *J. ACM*, 67(2), April 2020.

7 Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. New resolution-based QBF calculi and their proof complexity. *ACM Trans. Comput. Theory*, 11(4):26:1–26:42, 2019. doi:10.1145/3352155.

8 Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Understanding Cutting Planes for QBFs. In *FSTTCS 2016*, volume 65 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:15, 2016.

9 Robert Brummayer, Florian Lonsing, and Armin Biere. Automated testing and debugging of SAT and QBF solvers. In *SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2010.

10 Sravanthi Chede and Anil Shukla. Does QRAT simulate IR-calc? QRAT simulation algorithm for ∀Exp+Res cannot be lifted to IR-calc. *Electron. Colloquium Comput. Complex.*, page 104, 2021.

11 Hubie Chen. Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness. In *ICALP 2016*, pages 94:1–94:14, 2016.

12 Leroy Chew. Hardness and optimality in QBF proof systems modulo NP. In *SAT 2021*, pages 98–115, Cham, 2021. Springer.

13 Leroy Chew and Marijn Heule. Relating existing powerful proof systems for QBF. *Electron. Colloquium Comput. Complex.*, 27:159, 2020. URL: `https://eccc.weizmann.ac.il/report/2020/159`.

14 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

15 Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt Jr., Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. In *CADE 2017*, volume 10395 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2017.

16 Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *LPAR 2013*, pages 291–308. Springer, 2013. `doi:10.1007/978-3-642-45221-5_21`.

17 Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In Toby Walsh, editor, *IJCAI 2011*, pages 546–553. IJCAI/AAAI, 2011. URL: `http://ijcai.org/papers11/Papers/IJCAI11-099.pdf`.

18 Marijn J. H. Heule, Martina Seidl, and Armin Biere. Solution validation and extraction for QBF preprocessing. *J. Autom. Reason.*, 58(1):97–125, 2017.

19 Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

20 Mikolás Janota and João Marques-Silva. Solving QBF by clause selection. In *IJCAI 2015*, pages 325–331. AAAI Press, 2015.

21 Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015.

22 Toni Jussila, Armin Biere, Carsten Sinz, Daniel Kröning, and Christoph M. Wintersteiger. A first step towards a unified proof checker for QBF. In *SAT 2007*, pages 201–214, 2007. `doi:10.1007/978-3-540-72788-0_21`.

23 Benjamin Kiesl, Marijn J. H. Heule, and Martina Seidl. A little blocked literal goes a long way. In *SAT 2017*, volume 10491 of *Lecture Notes in Computer Science*, pages 281–297. Springer, 2017. `doi:10.1007/978-3-319-66263-3_18`.

24 Benjamin Kiesl and Martina Seidl. QRAT polynomially simulates ∀Exp+Res. In *SAT 2019*, volume 11628 of *Lecture Notes in Computer Science*, pages 193–202. Springer, 2019. `doi:10.1007/978-3-030-24258-9_13`.

25 Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995. `doi:10.1006/inco.1995.1025`.

26 Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019.

27 Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.

28 Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36:29–46, 1990.

29 Florian Lonsing and Armin Biere. Integrating dependency schemes in search-based QBF solvers. In *SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2010.

30 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019.

31 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-Resolution with dependency schemes. *J. Autom. Reason.*, 63(1):127–155, 2019.

32 Markus N Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In *FMCAD 2015*, pages 136–143. FMCAD Inc, 2015.

33 Robert A. Reckhow. *On the lengths of proofs in the propositional calculus.* PhD thesis, University of Toronto, 1976.

34 Matthias Schlaipfer, Friedrich Slivovsky, Georg Weissenbacher, and Florian Zuleger. Multi-linear strategy extraction for QBF expansion proofs via local soundness. In *SAT 2020*, volume 12178 of *Lecture Notes in Computer Science*, pages 429–446. Springer, 2020.

35 Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified Boolean formulas. In *ICTAI 2019*, pages 78–84. IEEE, 2019.

36 L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. *Proc. 5th ACM Symposium on Theory of Computing*, pages 1–9, 1973.

37 Martin Suda and Bernhard Gleiss. Local soundness for QBF calculi. In *SAT 2018*, volume 10929 of *Lecture Notes in Computer Science*, pages 217–234. Springer, 2018.

38 Allen Van Gelder. Contributions to the theory of practical quantified Boolean formula solving. In *Principles and Practice of Constraint Programming*, pages 647–663. Springer, 2012.

39 Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014.

40 Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *ICCAD 2002*, pages 442–449, 2002.

A Appendix

A.1 Proof of Simulation of IR-calc

▶ **Lemma 8.** *For* $0 < j \leq m$ *the following propositions have short derivations in Extended Frege:*

- $\mathrm{Dif}^j_L \rightarrow \bigvee_{i=1}^{j} \mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L$
- $\mathrm{Dif}^j_R \rightarrow \bigvee_{i=1}^{j} \mathrm{Dif}^i_R \wedge \neg \mathrm{Dif}^{i-1}_R$
- $\neg \mathrm{Eq}^j_{f=g} \rightarrow \bigvee_{i=1}^{j} \neg \mathrm{Eq}^i_{f=g} \wedge \mathrm{Eq}^{i-1}_{f=g}$. *For* $f, g \in \{L, R, \tau\}$.

Proof.

Induction Hypothesis on j. $\mathrm{Dif}^j_L \rightarrow \bigvee_{i=1}^{j} \mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L$ has an $O(j)$-size proof.

Base Case $j = 1$. $\mathrm{Dif}^1_L \rightarrow \mathrm{Dif}^1_L$ is a basic tautology that Frege can handle, Dif^0_L is false by definition so Frege can assemble $\mathrm{Dif}^1_L \rightarrow \mathrm{Dif}^1_L \wedge \neg \mathrm{Dif}^0_L$.

Inductive Step $j + 1$. $\neg \mathrm{Dif}^j_L \vee \mathrm{Dif}^j_L$ and $\mathrm{Dif}^{j+1}_L \rightarrow \mathrm{Dif}^{j+1}_L$ are tautologies that Frege can handle. Putting them together we get $\mathrm{Dif}^{j+1}_L \rightarrow \mathrm{Dif}^{j+1}_L \wedge (\neg \mathrm{Dif}^j_L \vee \mathrm{Dif}^j_L)$ and weaken to $\mathrm{Dif}^{j+1}_L \rightarrow (\mathrm{Dif}^{j+1}_L \wedge \neg \mathrm{Dif}^j_L) \vee \mathrm{Dif}^j_L$. Using the induction hypothesis, $\mathrm{Dif}^j_L \rightarrow \bigvee_{i=1}^{j} \mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L$, we can change this tautology to

$$\mathrm{Dif}^{j+1}_L \rightarrow (\mathrm{Dif}^{j+1}_L \wedge \neg \mathrm{Dif}^j_L) \vee \bigvee_{i=1}^{j} \mathrm{Dif}^i_L \wedge \neg \mathrm{Dif}^{i-1}_L$$

Note that since $\neg \mathrm{Dif}^0_R, \mathrm{Eq}^0_{L=\tau \sqcup \xi}, \mathrm{Eq}^0_{L=\tau \sqcup \sigma}$ are all true . The proofs for Dif^j_R, $\neg \mathrm{Eq}^j_{L=\tau \sqcup \sigma}$ and $\neg \mathrm{Eq}^j_{R=\tau \sqcup \xi}$ are identical modulo the variable names. ◀

▶ **Lemma 9.** *For* $0 \leq i \leq j \leq m$ *the following propositions that describe the monotonicity of* Dif *have short derivations in Extended Frege:*

- $\mathrm{Dif}^i_L \rightarrow \mathrm{Dif}^j_L$
- $\mathrm{Dif}^i_R \rightarrow \mathrm{Dif}^j_R$
- $\neg\,\mathrm{Eq}^i_{f=g} \rightarrow \neg\,\mathrm{Eq}^j_{f=g}$

Proof. For Dif_L and Dif_R,

Induction Hypothesis on j. $\mathrm{Dif}^i_L \rightarrow \mathrm{Dif}^j_L$ has an $O(j)$ proof.

Base Case $j = i$. $\mathrm{Dif}^i_L \rightarrow \mathrm{Dif}^i_L$ is a tautology that Frege can handle.

Inductive Step $j + 1$. $\mathrm{Dif}^{j+1}_L := \mathrm{Dif}^j_L \vee A$ where expression A depends on the domain of u_{j+1}. Therefore in all cases $\mathrm{Dif}^j_L \rightarrow \mathrm{Dif}^{j+1}_L$ is a straightforward corollary in Frege. Using the induction hypothesis $\mathrm{Dif}^i_L \rightarrow \mathrm{Dif}^j_L$ we can get $\mathrm{Dif}^i_L \rightarrow \mathrm{Dif}^{j+1}_L$. The proof is symmetric for R.

For $\neg\,\mathrm{Eq}_{f=g}$,

Induction Hypothesis on j. $\neg\,\mathrm{Eq}^i_{f=g} \rightarrow \neg\,\mathrm{Eq}^j_{f=g}$ has an $O(j)$ proof.

Base Case $j = i$. $\neg\,\mathrm{Eq}^i_{f=g} \rightarrow \neg\,\mathrm{Eq}^i_{f=g}$ is a tautology that Frege can handle.

Inductive Step $j + 1$. $\mathrm{Eq}^{j+1}_{f=g} := \mathrm{Eq}^j_{f=g} \wedge A$ where expression A depends on the domain of u_{j+1}. Therefore in all cases $\neg\,\mathrm{Eq}^j_{f=g} \rightarrow \neg\,\mathrm{Eq}^{j+1}_{f=g}$ is a straightforward corollary in Frege. Using the induction hypothesis $\neg\,\mathrm{Eq}^i_{f=g} \rightarrow \neg\,\mathrm{Eq}^j_{f=g}$ we can get $\neg\,\mathrm{Eq}^i_{f=g} \rightarrow \neg\,\mathrm{Eq}^{j+1}_{f=g}$. ◀

▶ **Lemma 10.** *For* $0 \leq i \leq j \leq m$ *the following propositions describe the relationships between the different extension variables.*

- $\mathrm{Eq}^i_{L=\tau} \rightarrow \neg\,\mathrm{Dif}^i_L$
- $\mathrm{Dif}^i_L \wedge \neg\,\mathrm{Dif}^{i-1}_L \rightarrow \mathrm{Eq}^{i-1}_{R=\tau}$
- $\mathrm{Dif}^i_L \wedge \neg\,\mathrm{Dif}^{i-1}_L \rightarrow \neg\,\mathrm{Dif}^{i-1}_R$
- $\mathrm{Eq}^i_{R=\tau} \rightarrow \neg\,\mathrm{Dif}^i_R$
- $\mathrm{Dif}^i_R \wedge \neg\,\mathrm{Dif}^{i-1}_R \rightarrow \mathrm{Eq}^{i-1}_{L=\tau}$
- $\mathrm{Dif}^i_R \wedge \neg\,\mathrm{Dif}^{i-1}_R \rightarrow \neg\,\mathrm{Dif}^{i-1}_L$

Proof.

Induction Hypothesis on i. $\mathrm{Eq}^i_{L=\tau} \rightarrow \neg\,\mathrm{Dif}^i_L$ in an $O(i)$-size eFrege proof.

Base Case $i = 0$. Dif^i_L is defined as 0 so $\neg\,\mathrm{Dif}^i_L$ is true and trivially implied by $\mathrm{Eq}^i_{L=\tau}$. Frege can manage this.

Inductive Step $i + 1$. If Set^{i+1}_τ is false then $\mathrm{Eq}^{i+1}_{L=\tau}$ is equivalent to $\mathrm{Eq}^i_{L=\tau} \wedge \neg\,\mathrm{Set}^{i+1}_L$ and $\neg\,\mathrm{Dif}^{i+1}_L$ is equivalent to $\neg\,\mathrm{Dif}^i_L \wedge \neg\,\mathrm{Set}^{i+1}_L \vee \neg\,\mathrm{Eq}^i_{L=\tau}$. If Set^{i+1}_τ is true then $\mathrm{Eq}^{i+1}_{L=\tau}$ is equivalent to $\mathrm{Eq}^i_{L=\tau} \wedge \mathrm{Set}^{i+1}_L \wedge (\mathrm{Val}^{i+1}_L \leftrightarrow \mathrm{Val}^{i+1}_\tau)$ and $\neg\,\mathrm{Dif}^{i+1}_L$ is equivalent to $\neg\,\mathrm{Dif}^i_L \wedge \mathrm{Set}^{i+1}_L \wedge (\mathrm{Val}^{i+1}_L \leftrightarrow \mathrm{Val}^{i+1}_\tau) \vee \neg\,\mathrm{Eq}^i_{L=\tau}$. Therefore using the induction hypothesis $\mathrm{Eq}^i_{L=\tau} \rightarrow \neg\,\mathrm{Dif}^i_L$. Similarly for R.

The formulas $\mathrm{Dif}^i_L \wedge \neg\,\mathrm{Dif}^{i-1}_L \rightarrow \mathrm{Eq}^{i-1}_{R=\tau}$ are simple corollaries of the inductive definition of Dif^i_L, and combined with $\mathrm{Eq}^{i-1}_{R=\tau} \rightarrow \neg\,\mathrm{Dif}^{i-1}_R$ we get $\mathrm{Dif}^i_L \wedge \neg\,\mathrm{Dif}^{i-1}_L \rightarrow \neg\,\mathrm{Dif}^{i-1}_R$. Similarly if we swap L and R. ◀

▶ **Lemma 11.** *For any* $0 \leq i \leq m$ *the following propositions are true and have short Extended Frege proofs.*

- $L \wedge \mathrm{Dif}^i_L \rightarrow \neg\,\mathrm{anno}_{x,L}(\tau)$
- $R \wedge \mathrm{Dif}^i_R \rightarrow \neg\,\mathrm{anno}_{x,R}(\tau)$

Proof. We primarily use the disjunction in Lemma 8 $\mathrm{Dif}^i_L \rightarrow \bigvee_{i=1}^{j} \mathrm{Dif}^i_L \wedge\neg \mathrm{Dif}^{i-1}_L$.

In each disjunct $\mathrm{Dif}^i_L \wedge\neg \mathrm{Dif}^{i-1}_L$ we can say that the difference triggers at that point. We can represent that in a proposition that can be proven in eFrege: $\mathrm{Dif}^i_L \wedge\neg \mathrm{Dif}^{i-1}_L \rightarrow ((\mathrm{Set}^i_L \oplus \mathrm{Set}^i_\tau) \vee (\mathrm{Set}^i_\tau \wedge(\mathrm{Val}^i_L \oplus \mathrm{Val}^i_\tau)))$ If L differs from τ on a Set^i_L value we contradict $\mathrm{anno}_{x,L}(\tau)$ in one of two ways: $L \wedge (\mathrm{Set}^i_L \oplus \mathrm{Set}^i_\tau) \wedge \mathrm{Set}^i_L \rightarrow \neg \mathrm{Set}^i_\tau$ or $L \wedge (\mathrm{Set}^i_L \oplus \mathrm{Set}^i_\tau) \wedge \neg \mathrm{Set}^i_L \rightarrow \mathrm{Set}^i_\tau$.

If L differs from τ on a Val^i_L value when $\mathrm{Set}^i_L = \mathrm{Set}^i_\tau = 1$ we contradict $\mathrm{anno}_{x,L}(\tau)$ in one of two ways:

- $L \wedge \mathrm{Set}^i_L \wedge \mathrm{Set}^i_\tau \wedge(\mathrm{Set}^i_\tau \rightarrow (\mathrm{Val}^i_L \oplus \mathrm{Val}^i_\tau)) \wedge \mathrm{Val}^i_L \rightarrow \neg \mathrm{Val}^i_\tau \wedge u_i$
- $L \wedge \mathrm{Set}^i_L \wedge \mathrm{Set}^i_\tau \wedge(\mathrm{Set}^i_\tau \rightarrow (\mathrm{Val}^i_L \oplus \mathrm{Val}^i_\tau)) \wedge \neg \mathrm{Val}^i_L \rightarrow \mathrm{Val}^i_\tau \wedge\neg u_i$.

When put together with the big disjunction this lends itself to a short eFrege proof which is also symmetric for R. ◀

▶ **Lemma 12.** *For any $1 \leq j \leq m$ the following propositions are true and have a short Extended Frege proof.*

- $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_L$
- $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_R$
- $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow (\mathrm{Set}^j_B \leftrightarrow \mathrm{Set}^j_L)$
- $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Set}^i_B \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_L)$
- $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow (\mathrm{Set}^j_B \leftrightarrow \mathrm{Set}^j_R)$
- $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Set}^i_B \rightarrow (\mathrm{Val}^j_B \leftrightarrow \mathrm{Val}^j_R)$

Proof. We first show $\neg \mathrm{Eq}^j_{L=\tau} \rightarrow \neg \mathrm{Eq}^{j-1}_{R=\tau} \vee \mathrm{Dif}^j_L \vee \mathrm{Dif}^j_R$ and $\neg \mathrm{Eq}^j_{R=\tau} \rightarrow \neg \mathrm{Eq}^{j-1}_L \vee \mathrm{Dif}^j_R \vee \mathrm{Dif}^j_R$. $\neg \mathrm{Eq}^{j-1}_{R=\tau}$ and $\neg \mathrm{Eq}^{j-1}_{L=\tau}$ are the problems here respectively, but they can be removed via induction to eventually get $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_L$ and $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_{R=\tau}$. The remaining implications are corollaries of these and rely on the definition of Eq, Set_B and Val_B.

Induction Hypothesis on j. $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_L$ and $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_R$.

Base Case $j = 0$. $\mathrm{Eq}^j_{L=\tau}$ and $\mathrm{Eq}^j_{R=\tau}$ are both true by definition so the implications automatically hold.

Inductive Step j. $\neg \mathrm{Eq}^{j+1}_{L=\tau} \rightarrow \neg \mathrm{Eq}^{j-1}_{L=\tau} \vee(\mathrm{Set}^j_L \oplus \mathrm{Set}^j_\tau) \vee (\mathrm{Set}^j_L \wedge(\mathrm{Val}^j_L \oplus \mathrm{Val}^j_\tau))$, $(\mathrm{Set}^j_L \oplus \mathrm{Set}^j_\tau) \vee (\mathrm{Set}^j_L \wedge(\mathrm{Val}^j_L \oplus \mathrm{Val}^j_\tau)) \rightarrow \mathrm{Dif}^j_L \vee\neg \mathrm{Eq}^{j-1}_{R=\tau}$ so we get $\neg \mathrm{Eq}^j_{L=\tau} \rightarrow \neg \mathrm{Eq}^{j-1}_{L=\tau} \vee \mathrm{Dif}^j_L \vee\neg \mathrm{Eq}^{j-1}_{R=\tau}$, which using the induction hypothesis can be generalised to $\neg \mathrm{Eq}^j_{L=\tau} \rightarrow \mathrm{Dif}^j_R \vee \mathrm{Dif}^j_L$ which is equivalent to $\neg \mathrm{Dif}^j_L \wedge\neg \mathrm{Dif}^j_R \rightarrow \mathrm{Eq}^j_L$. Similarly when swapping L and R.

We can obtain the remaining propositions as corollaries by using the definition of Eq. ◀

Nonetheless, Dif^i_L and Dif^i_R still end up being relevant to the choice of Val^j_B.

▶ **Lemma 13.** *For any $0 \leq i \leq m$ the following propositions are true and have short Extended Frege proofs.*

- $\mathrm{Dif}^i_L \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_L) \wedge (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_L)$
- $\neg \mathrm{Dif}^i_L \wedge \mathrm{Dif}^i_R \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_R) \wedge (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_R)$

Proof. Suppose we want to prove $\mathrm{Dif}^i_L \rightarrow (\mathrm{Val}^i_B \leftrightarrow \mathrm{Val}^i_L) \wedge (\mathrm{Set}^i_B \leftrightarrow \mathrm{Set}^i_L)$. We will assume the definition

$$\mathrm{Dif}^i_L := \mathrm{Dif}^{i-1}_L \vee(\mathrm{Eq}^{i-1}_R \wedge((\mathrm{Set}^i_L \oplus \mathrm{Set}^i_\tau) \vee (\mathrm{Set}^i_\tau \wedge(\mathrm{Val}^i_L \oplus \mathrm{Val}^i_\tau))))$$

and show that following proposition is falsified

$$\neg\,\mathrm{Dif}_L^{i-1} \wedge (\mathrm{Dif}_R^{i-1} \vee (\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i) \vee (\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)))$$

The first thing is that we only need to consider $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1}$ as Dif_L^{i-1} already falsifies our proposition. Next we show $\neg\,\mathrm{Dif}_R^{i-1}$ is forced to be true in this situation. To do this we need Lemma 10 for $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1} \rightarrow \neg\,\mathrm{Dif}_R^{i-1}$.

Now we use $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1} \rightarrow ((\mathrm{Set}_L^i \oplus \mathrm{Set}_\tau^i) \vee (\mathrm{Set}_\tau^i \wedge (\mathrm{Val}_L^i \oplus \mathrm{Val}_\tau^i)))$, we break this down into three cases

1. $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1} \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_\tau^i$
2. $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1} \wedge \mathrm{Set}_L^i \wedge \neg\,\mathrm{Set}_\tau^i$
3. $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1} \wedge (\mathrm{Set}_\tau^i \wedge (\mathrm{Val}_L^i \oplus \mathrm{Val}_\tau^i))$

1. $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1}$ contradicts Dif_R^{i-1}, Set_τ^i contradicts $(\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i)$, and $\neg\,\mathrm{Set}_L^i$ contradicts $(\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i))$.
2. $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1}$ contradicts Dif_R^{i-1}, Set_L^i contradicts $(\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i)$, and $\neg\,\mathrm{Set}_\tau^i$ contradicts $(\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i))$.
3. $\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1}$ contradicts Dif_R^{i-1}, Set_τ^i contradicts $(\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i)$, $(\mathrm{Val}_L^i \oplus \mathrm{Val}_\tau^i)$ contradicts $(\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i))$

Since in all cases we contradict$\neg\,\mathrm{Dif}_L^{i-1} \wedge (\mathrm{Dif}_R^{i-1} \vee (\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i) \vee (\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)))$ then as per definition $(\mathrm{Val}_B, \mathrm{Set}_B) = (\mathrm{Val}_L, \mathrm{Set}_L)$. Using $\mathrm{Dif}_L^i \rightarrow (\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_L^{i-1}) \vee \mathrm{Dif}_L^{i-1}$ we get $\mathrm{Dif}_L^i \rightarrow (\mathrm{Val}_B^i \leftrightarrow \mathrm{Val}_L^i) \wedge (\mathrm{Set}_B^i \leftrightarrow \mathrm{Set}_L^i)$, in a polynomial number of Frege lines.

Now we suppose we want to prove the second proposition $\neg\,\mathrm{Dif}_L^i \wedge \mathrm{Dif}_R^i \rightarrow (\mathrm{Val}_B^i \leftrightarrow \mathrm{Val}_R^i) \wedge (\mathrm{Set}_B^i \leftrightarrow \mathrm{Set}_R^i)$. We need $\neg\,\mathrm{Dif}_L^i \wedge \mathrm{Dif}_R^i$ to satisfy $\neg\,\mathrm{Dif}_L^{i-1} \wedge (\mathrm{Dif}_R^{i-1} \vee (\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i) \vee (\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)))$

Lemma gives us that $\neg\,\mathrm{Dif}_L^i \rightarrow \neg\,\mathrm{Dif}_L^{i-1}$. We can show that $\neg\,\mathrm{Dif}_L^{i-1} \wedge \neg\,\mathrm{Dif}_R^{i-1} \rightarrow \mathrm{Eq}_{L=\tau}^{i-1}$ using Lemma 15. This allows us to examine just the part where the difference is being triggered $\neg\,\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_R^{i-1} \rightarrow (\mathrm{Set}_\tau^i \leftrightarrow \mathrm{Set}_L^i) \wedge (\mathrm{Set}_\tau^i \rightarrow (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i))$.

Suppose the term $(\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i)$ is false, assuming Dif_R^{i-1} is also false, we have to show that $(\mathrm{Set}_\tau^i \wedge \mathrm{Set}_L^i \wedge (\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)$ will be satisfied. We look at the three ways the term $(\neg\,\mathrm{Set}_\tau^i \wedge \neg\,\mathrm{Set}_L^i \wedge \mathrm{Set}_R^i)$ can be falsified and show that all the parts of the remaining term must be satisfied when assuming $\neg\,\mathrm{Dif}_L^i \wedge \mathrm{Dif}_R^i \wedge \neg\,\mathrm{Dif}_R^{i-1}$

1. Set_τ^i, in this case $(\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)$ is active and Set_L^i is implied by $(\mathrm{Set}_\tau^i \leftrightarrow \mathrm{Set}_L^i)$.
2. Set_L^i, Set_τ^i is implied by $(\mathrm{Set}_\tau^i \leftrightarrow \mathrm{Set}_L^i)$, then $(\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)$ is active.
3. $\neg\,\mathrm{Set}_R^i$, then using Dif_R^i and $\neg\,\mathrm{Dif}_R^{i-1}$ we must Set_τ^i (as this is the only allowed way Dif can trigger). Once again, $(\mathrm{Val}_\tau^i \leftrightarrow \mathrm{Val}_L^i)$ is active and Set_L^i is implied by $(\mathrm{Set}_\tau^i \leftrightarrow \mathrm{Set}_L^i)$

Since our trigger formula is always satisfied when $\neg\,\mathrm{Dif}_L^i \wedge \mathrm{Dif}_R^i \wedge \neg\,\mathrm{Dif}_R^{i-1}$. It means that $(\mathrm{Val}_B, \mathrm{Set}_B) = (\mathrm{Val}_R, \mathrm{Set}_R)$. Using $\mathrm{Dif}_R^i \rightarrow (\mathrm{Dif}_R^i \wedge \neg\,\mathrm{Dif}_R^{i-1}) \vee \mathrm{Dif}_R^{i-1}$ we get $\neg\,\mathrm{Dif}_L^i \wedge \mathrm{Dif}_R^i \rightarrow (\mathrm{Val}_B^i \leftrightarrow \mathrm{Val}_R^i) \wedge (\mathrm{Set}_B^i \leftrightarrow \mathrm{Set}_R^i)$, in a polynomial number of Frege lines. ◀

▶ **Lemma 14.** *The following propositions are true and have short Extended Frege proofs.*

- $B \wedge \mathrm{Dif}_L^m \rightarrow B_L$
- $B \wedge \neg\,\mathrm{Dif}_L^m \wedge \mathrm{Dif}_R^m \rightarrow B_R$

Proof. We use the disjunction $\mathrm{Dif}_L^m \rightarrow \bigvee_{j=1}^m \mathrm{Dif}_L^j \vee \neg\,\mathrm{Dif}_L^{j-1}$ So there is some j where this is the case.

- For $1 \leq i < j$ observe that $\mathrm{Dif}_L^j \vee \neg\,\mathrm{Dif}_L^{j-1} \rightarrow \neg\,\mathrm{Dif}_R^{j-1}$. Now these negative literals propagate downwards. $\neg\,\mathrm{Dif}_L^{j-1} \wedge \neg\,\mathrm{Dif}_R^{j-1} \rightarrow \neg\,\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_R^i$ for $0 \leq i < j$ and $\neg\,\mathrm{Dif}_L^i \wedge \neg\,\mathrm{Dif}_R^i$ means that B and L are consistent for those i as proven in Lemma 12.

- For $j \leq k \leq m$, $\mathrm{Dif}^j_L \rightarrow \mathrm{Dif}^k_L$ and Dif^k_L means B and L are consistent on those k as proven in Lemma 13.
- For indices greater than m, $B \wedge \mathrm{Dif}^m_L$ falsifies $\neg \mathrm{Dif}^m_L \wedge (\mathrm{Dif}^m_R \vee \bar{x})$, so B and L are consistent on those indices.

With the second proposition $\mathrm{Dif}^m_R \rightarrow \bigvee_{j=1}^m \mathrm{Dif}^j_R \vee \neg \mathrm{Dif}^{j-1}_R$ once again. So there is some j where this is the case. Note that $\neg \mathrm{Dif}^m_L \rightarrow \neg \mathrm{Dif}^k_L$ for $k \leq m$.

- For $1 \leq i < j$, both $\neg \mathrm{Dif}^i_L$ and $\neg \mathrm{Dif}^i_R$ occur so then B and R are consistent for these values.
- For $j \leq k \leq m$, $\mathrm{Dif}^j_R \rightarrow \mathrm{Dif}^k_R$ and $\mathrm{Dif}^k_R \wedge \neg \mathrm{Dif}^k_L$ means B and R are consistent on those k as proven in Lemma 13.
- For indices greater than m, $B \wedge \mathrm{Dif}^m_R \wedge \neg \mathrm{Dif}^m_L$ satisfies $\neg \mathrm{Dif}^m_L \wedge (\mathrm{Dif}^m_R \vee \bar{x})$, so B and R are consistent on those indices. ◀

▶ **Lemma 15.** *The following propositions are true and have short Extended Frege proofs.*

- $B \wedge \neg \mathrm{Dif}^m_L \wedge \neg \mathrm{Dif}^m_R \rightarrow B_L \vee \neg x$
- $B \wedge \neg \mathrm{Dif}^m_L \wedge \neg \mathrm{Dif}^m_R \rightarrow B_R \vee x$

Proof. For indices $1 \leq i \leq m$, but since $\neg \mathrm{Dif}^m_L \rightarrow \neg \mathrm{Dif}^i_L$ and $\neg \mathrm{Dif}^m_R \rightarrow \neg \mathrm{Dif}^i_R$, Lemma 12 can be used to show that $B \wedge \mathrm{Dif}^m_L \wedge \mathrm{Dif}^m_R$ leads to $\mathrm{Set}^i_B = \mathrm{Set}^i_L = \mathrm{Set}^i_R$ and $\mathrm{Val}^i_B = \mathrm{Val}^i_L = \mathrm{Val}^i_R$ whenever Set^i_B is also true. Extended Frege can prove $O(m)$ many propositions expressing as such.

For $i > m$, by definition $B \wedge \neg \mathrm{Dif}^m_L \wedge \neg \mathrm{Dif}^m_R \wedge x$ gives $\mathrm{Set}^i_B = \mathrm{Set}^i_L$ and $\mathrm{Val}^i_B = \mathrm{Val}^i_L$. And $B \wedge \neg \mathrm{Dif}^m_L \wedge \neg \mathrm{Dif}^m_R \wedge \neg x$ gives $\mathrm{Set}^i_B = \mathrm{Set}^i_R$ and $\mathrm{Val}^i_B = \mathrm{Val}^i_R$. The sum of this is that $B \wedge \mathrm{Dif}^m_L \wedge \mathrm{Dif}^m_R \wedge x \rightarrow B_L$ and $B \wedge \mathrm{Dif}^m_L \wedge \mathrm{Dif}^m_R \wedge \neg x \rightarrow B_R$. ◀

A.2 Local Strategy Extraction for Simulation of IRM-calc

A.2.1 Conversion

$\mathrm{anno}_{x,S}(\tau) = \bigwedge_{1/u_i \in \tau} (\mathrm{Set}^i_S \wedge u_i) \wedge \bigwedge_{0/u_i \in \tau} (\mathrm{Set}^i_S \wedge \bar{u}_i) \wedge \bigwedge_{*/u_i \in \tau} (\mathrm{Set}^i_S) \wedge \bigwedge_{u_i \notin \mathsf{dom}(\tau)} (\neg \mathrm{Set}^i_S)$.
$\mathrm{con}_S(x^\tau) = x \wedge \mathrm{anno}_{x,S}(\tau)$, $\mathrm{con}_S(C_1) = \bigvee_{x^\tau \in C_1} \mathrm{con}(x^\tau)$

A.2.2 Equivalence

$\mathrm{Eq}^0_{f=g} := 1, \quad \mathrm{Eq}^i_{f=g} := \mathrm{Eq}^{i-1}_{f=g} \wedge (\mathrm{Set}^i_f)$ when $*/u_i \in g$
$\mathrm{Eq}^i_{f=g} := \mathrm{Eq}^{i-1}_{f=g} \wedge (\mathrm{Set}^i_f \leftrightarrow \mathrm{Set}^i_g) \wedge (\mathrm{Set}^i_f \rightarrow (\mathrm{Val}^i_f \leftrightarrow \mathrm{Val}^i_g))$ when $*/u_i \notin g$

A.2.3 Difference

$\mathrm{Dif}^0_L := 0$ and $\mathrm{Dif}^0_R := 0$
For $u_i \notin \mathsf{dom}(\tau \sqcup \sigma \sqcup \xi)$, $\mathrm{Dif}^i_L := \mathrm{Dif}^{i-1}_L \vee (\mathrm{Eq}^{i-1}_{R=\tau \sqcup \xi} \wedge (\mathrm{Set}^i_L)$,
For $u_i \in \mathsf{dom}(\tau)$, $\mathrm{Dif}^i_L := \mathrm{Dif}^{i-1}_L \vee (\mathrm{Eq}^{i-1}_{R=\tau \sqcup \xi} \wedge (\neg \mathrm{Set}^i_L \vee (\mathrm{Set}^i_\tau \wedge (\mathrm{Val}^i_L \oplus \mathrm{Val}^i_\tau))))$
For $u_i \in \mathsf{dom}(\sigma)$, $\mathrm{Dif}^i_L := \mathrm{Dif}^{i-1}_L \vee (\mathrm{Eq}^{i-1}_{R=\tau \sqcup \xi} \wedge (\neg \mathrm{Set}^i_L)$
For $u_i \in \mathsf{dom}(\xi)$, $\mathrm{Dif}^i_L := \mathrm{Dif}^{i-1}_L \vee (\mathrm{Eq}^{i-1}_{R=\tau \sqcup \xi} \wedge (\mathrm{Set}^i_L)$

A.2.4 Policy Variables

We define the policy variables Val^i_B and Set^i_B based on a number of cases, in all cases Val^i_B and Set^i_B are defined on variables left of u_i.

For $u_i \notin \mathsf{dom}(\tau \sqcup \sigma \sqcup \xi)$, $u_i < x$,

$$(\mathrm{Val}^i_B, \mathrm{Set}^i_B) = \begin{cases} (\mathrm{Val}^i_R, \mathrm{Set}^i_R) & \text{if } \neg \mathrm{Dif}^{i-1}_L \wedge (\mathrm{Dif}^{i-1}_R \vee \neg \mathrm{Set}^i_L) \\ (\mathrm{Val}^i_L, \mathrm{Set}^i_L) & \text{otherwise.} \end{cases}$$

For $u_i \in \mathsf{dom}(\tau)$,

$$(\mathrm{Val}^i_B, \mathrm{Set}^i_B) = \begin{cases} (\mathrm{Val}^i_R, \mathrm{Set}^i_R) & \text{if } \neg \mathrm{Dif}^{i-1}_L \wedge (\mathrm{Dif}^{i-1}_R \vee (\mathrm{Set}^i_L \wedge (\mathrm{Val}^i_L \leftrightarrow \mathrm{Val}^i_\tau))) \\ (\mathrm{Val}^i_L, \mathrm{Set}^i_L) & \text{otherwise.} \end{cases}$$

For $*/u_i \in \sigma$,

$$(\mathrm{Val}^i_B, \mathrm{Set}^i_B) = \begin{cases} (0, 1) & \text{if } \neg \mathrm{Dif}^{i-1}_L \wedge \mathrm{Dif}^{i-1}_R \wedge \neg \mathrm{Set}^i_R \\ (\mathrm{Val}^i_R, \mathrm{Set}^i_R) & \text{if } \neg \mathrm{Dif}^{i-1}_L \wedge \mathrm{Set}^i_R \wedge (\mathrm{Dif}^{i-1}_R \vee \mathrm{Set}^i_L) \\ (\mathrm{Val}^i_L, \mathrm{Set}^i_L) & \text{otherwise.} \end{cases}$$

For $*/u_i \in \xi$,

$$(\mathrm{Val}^i_B, \mathrm{Set}^i_B) = \begin{cases} (0, 1) & \text{if } \mathrm{Dif}^{i-1}_L \wedge \neg \mathrm{Set}^i_L \\ (\mathrm{Val}^i_R, \mathrm{Set}^i_R) & \text{if } \neg \mathrm{Dif}^{i-1}_L \wedge (\mathrm{Dif}^{i-1}_R \vee \neg \mathrm{Set}^i_L) \\ (\mathrm{Val}^i_L, \mathrm{Set}^i_L) & \text{otherwise.} \end{cases}$$

For $u_i > x$,

$$(\mathrm{Val}^i_B, \mathrm{Set}^i_B) = \begin{cases} (\mathrm{Val}^i_R, \mathrm{Set}^i_R) & \text{if } \neg \mathrm{Dif}^m_L \wedge (\mathrm{Dif}^m_R \vee \neg x) \\ (\mathrm{Val}^i_L, \mathrm{Set}^i_L) & \text{otherwise.} \end{cases}$$

Blazing a Trail via Matrix Multiplications: A Faster Algorithm for Non-Shortest Induced Paths

Yung-Chung Chiu
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan

Hsueh-I Lu ✉
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan

Abstract

For vertices u and v of an n-vertex graph G, a *uv-trail* of G is an induced uv-path of G that is not a shortest uv-path of G. Berger, Seymour, and Spirkl [*Discrete Mathematics* 2021] gave the previously only known polynomial-time algorithm, running in $O(n^{18})$ time, to either output a uv-trail of G or ensure that G admits no uv-trail. We reduce the complexity to the time required to perform a poly-logarithmic number of multiplications of $n^2 \times n^2$ Boolean matrices, leading to a largely improved $O(n^{4.75})$-time algorithm.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Induced subgraph, induced path, non-shortest path, dynamic data structure

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.23

Funding *Hsueh-I Lu*: MOST grants 110–2221–E–002–075–MY3 and 107–2221–E–002–032–MY3.

1 Introduction

Let G be an n-vertex undirected and unweighted graph. Let $V(G)$ consist of the vertices of G. For any subgraph H of G, let $G[H]$ be the subgraph of G induced by $V(H)$. A subgraph H of G is *induced* if $G[H] = H$. That is, an induced subgraph of G is a subgraph of G that can be obtained by deleting a set of vertices together with its incident edges from G. Various kinds of induced subgraphs are involved in the deepest results of graph theory and graph algorithms. One of the most prominent examples concerns the *perfection* of G that the chromatic number of each induced subgraph H of G equals the clique number of H. A graph is *odd* (respectively, *even*) if it has an odd (respectively, even) number of edges. A *hole* of G is an induced cycle of G having at least four edges. The seminal Strong Perfect Graph Theorem of Chudnovsky, Robertson, Seymour, and Thomas [16, 21], conjectured by Berge in 1960 [4, 5, 6], ensures that the perfection of a graph G can be determined by detecting an odd hole in G or its complement $\bar{G}$. Based on the theorem, the first known polynomial-time algorithms for recognizing perfect graphs take $O(n^{18})$ [30] and $O(n^9)$ [13] time. The $O(n^9)$-time version can be implemented to run in $O(n^{8.373})$ time via Boolean matrix multiplications [52, §6.2].

Detecting a class of induced subgraphs can be much more difficult than detecting its counterpart that need not be induced. For instance, detecting a path spanning three prespecified vertices is tractable (via, e. g., [50, 58]), but the *three-in-a-path* problem that detects an induced path spanning three prespecified vertices is NP-hard (see, e. g., [43, 52]). Cycle detection has a similar situation. Detecting a cycle of length three, which has to be induced, is the classical triangle detection problem that can also be solved efficiently by Boolean matrix multiplications. Although it is tractable to detect a cycle of length at least four spanning two prespecified vertices (also via, e. g., [50, 58]), the *two-in-a-cycle* problem that detects a hole spanning two prespecified vertices is NP-hard (and so are the corresponding

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 23; pp. 23:1–23:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

one-in-an-even-cycle and one-in-an-odd-cycle problems) [8, 9]. See, e. g., [57, §3.1] for graph classes on which the two-in-a-cycle problem is tractable. Detecting a tree spanning an arbitrary set of prespecified vertices is easy via computing the connected components of G. Detecting an induced tree spanning an arbitrary set of prespecified vertices is NP-hard [42]. The *three-in-a-tree* problem that detects an induced tree spanning three prespecified vertices was first shown to be solvable in $O(n^4)$ time [20] and then in $O(n^2 \log^2 n)$ time [52]. The tractability of the corresponding k-in-a-tree problem for any fix $k \geq 4$ is unknown. See [54] for an $O(n^4)$-time algorithm for the k-in-a-tree problem in a graph of girth at least k.

As for subgraph detection without the requirement of spanning prespecified vertices, detecting a cycle is straightforward. Even and odd cycles are also long known to be efficiently detectable (see, e. g., [3, 32, 63]). While detecting a hole (i. e., recognizing chordal graphs) is solvable in $O(n^2)$ time [59, 60, 61], detecting an odd (respectively, even) hole is more difficult. There are early $O(n^3)$-time algorithms for detecting odd and even holes in planar graphs [46, 56], but the tractability of detecting an odd hole was open for decades (see, e. g., [22, 24, 27]) until the recent major breakthrough of Chudnovsky, Scott, Seymour, and Spirkl [19]. Their $O(n^9)$-time algorithm is later implemented to run in $O(n^8)$ time [52], immediately implying the currently best known algorithm for recognizing perfect graphs based on the Strong Perfect Graph Theorem. It is also recently known that a shortest odd hole, if any, can be found in $O(n^{14})$ time [18]. As for detecting even holes, the first polynomial-time algorithm, running in about $O(n^{40})$ time, appeared in 1997 [23, 25, 26]. It takes a line of intensive efforts to bring down the complexity to $O(n^{31})$ [14], $O(n^{19})$ [31], $O(n^{11})$ [10], and finally $O(n^9)$ [52]. The tractability of finding a shortest even hole, open for 16 years [14, 48], is resolved by a newly announced $O(n^{31})$-time algorithm [12]. See [17] (respectively, [29]) for detecting an odd (respectively, even) hole with a prespecified length lower bound. See [1, 15] for the first polynomial-time algorithm for finding an independent set of maximum weight in a graph having no hole of length at least five. See [33] for upper and lower bounds on the complexity of detecting an $O(1)$-vertex induced subgraph.

The *two-in-a-path* problem that detects an induced path spanning two prespecified vertices is equivalent to determining whether the two vertices are connected. On the other hand, the corresponding two-in-an-odd-path and two-in-an-even-path problems are NP-hard [8, 9], although each of them admits an $O(n^7)$-time algorithm when G is planar [49]. See [35, 37, 55] for how an induced even uv-path of G affects the perfection of G. See [51] for a conjecture by Erdős on how an induced uv-path of G affects the connectivity between u and v in G. Finding a longest uv-path in G that has to (respectively, need not) be induced is NP-hard [39, GT23] (respectively, [39, ND29]). See [41, 47] for longest or long induced paths in special graphs. The presence of long induced paths in G affects the tractability of coloring G [40]. See also [1] for the first polynomial-time algorithm for finding a minimum feedback vertex set of a graph having no induced path of length at least five. Detecting a non-shortest uv-path in G is easy. A k-th shortest uv-path in G can also be found in near linear time [34]. See [44] for listing induced paths and holes. See [11, §4] for the parameterized complexity of detecting an induced path with a prespecified length. Detecting an induced uv-path in a directed graph G is NP-complete (even if G is planar) [36] and $W[1]$-complete [43]. However, the tractability of detecting a non-shortest induced uv-path in an undirected graph G was unknown until the recent result of Berger, Seymour, and Spirkl [7].

Let $\|G\|$ denote the number of edges in G. A path with end-vertices u and v is a *uv-path*. If P is a path with $\{u, v\} \subseteq V(P)$, then let $P[u, v]$ denote the uv-path of P. A uv-path P of G is *shortest* if G admits no uv-path Q with $\|Q\| < \|P\|$, so each shortest uv-path of G is induced. We call an induced uv-path of G that is not a shortest uv-path of G a *uv-trail* of G.

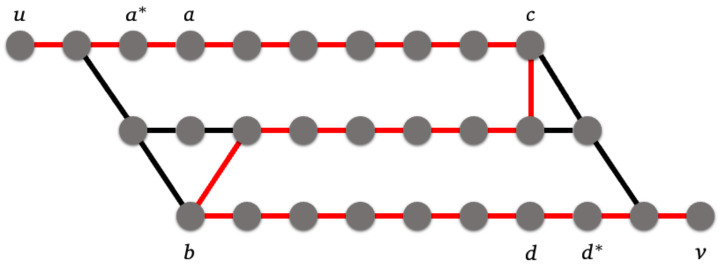


Figure 1 The red uv-path P is the only uv-trail of the uv-straight graph G. The twist pair of P is (c, b). The twist of P is 6. $P[a^*, c]$ and $P[b, d^*]$ form a pair of wings for the quadruple (a, b, c, d) of $V(G)$ in G.

See Figure 1 for an example. A graph admitting no uv-trail is *uv-trailless*. Berger, Seymour, and Spirkl [7] gave the formerly only known polynomial-time algorithm, running in $O(n^{18})$ time, to either output a uv-trail of G or ensure that G is uv-trailless. Their result leads to an $O(n^{21})$-time algorithm [28] to determine whether all holes of G have the same length. We improve the time of finding a uv-trail to $O(n^{4.75})$ as summarized in the following theorem, where the $\tilde{O}$ notation hides poly-logarithmic factors and $M(m) = O(m^{2.373})$ [2, 53, 62] denotes the time of multiplying two $m \times m$ Boolean matrices.

▶ **Theorem 1.** *For any two vertices u and v of an n-vertex graph G, it takes $\tilde{O}(M(n^2))$ time to either obtain a uv-trail of G or ensure that G is uv-trailless.*

Theorem 1 immediately reduces the time of recognizing a graph with all holes the same length from $O(n^{21})$ to $O(n^{7.75})$.

Technical overview

Berger et al.'s and our algorithms are based on the following "guess-and-verify" approach. A subroutine B taking an ℓ-tuple of $V(G)$ as the only argument is a *uv-trailblazer of degree ℓ* for G if running B on all ℓ-tuples of $V(G)$ always reports a uv-trail of G unless G is uv-trailless. We call an ℓ-tuple of $V(G)$ on which B reports a uv-trail of G a *trail marker* for B. An $O(f(n))$-time uv-trailblazer of degree ℓ for G immediately implies the following $O(n^\ell \cdot f(n))$-time *trailblazing algorithm* for G: Run B on each ℓ-tuple $(a_1, \ldots, a_\ell)$ of $V(G)$ to either obtain a uv-trail of G or ensure that $(a_1, \ldots, a_\ell)$ is not a trail marker for B. If none of the $O(n^\ell)$ iterations produces a uv-trail of G, then report that G is uv-trailless.

A graph H is *uv-straight* [7] if $\{u, v\} \subseteq V(H)$ and each vertex of H belongs to at least one shortest uv-path of H. For instance, the graph in Figure 1 is uv-straight. Berger et al.'s algorithm starts with an $O(n^3)$-time preprocessing step (see Lemma 2) that either reports a uv-trail of G or obtains a uv-straight graph H with $V(H) \subseteq V(G)$ such that (a) a uv-trail of G can be obtained from a uv-trail of H in $O(n^2)$ time and (b) if H is uv-trailless, then so is G. If no uv-trail is reported by the preprocessing, then the main procedure runs an $O(n^{18})$-time trailblazing algorithm on the uv-straight graph H based on an $O(n^4)$-time degree-14 uv-trailblazer for H. As for postprocessing, if a uv-trail of H is obtained by the main procedure, then report a uv-trail of G obtainable in $O(n^2)$ time as ensured by the preprocessing. Otherwise, report that G is uv-trailless.

Our $O(n^{4.75})$-time algorithm adopts the preprocessing and postprocessing steps of Berger et al., while reducing the preprocessing time from $O(n^3)$ to $O(M(n))$ (see Lemma 6). For the benefit of the main procedure, we run a second preprocessing step, taking $O(n^{4.75})$ time, to compute a static data structure from which a pair of "wings" that are some disjoint paths in H, if any, for each quadruple of $V(H)$ can be obtained in $O(n)$ time (see Lemma 7). Our main procedure is also a trailblazing algorithm, based on a faster uv-trailblazer of a much lower degree for H: We reduce the time from $O(n^4)$ to $O(n^2 \log^2 n)$ and largely bring down the degree from 14 to 2. Thus, the main procedure runs in $O(n^4 \cdot \log^2 n)$ time, even faster than the second preprocessing step.

The key to our improved uv-trailblazer is a new observation, described by Lemma 5, on any shortest uv-trail P of a uv-straight graph G. Specifically, Berger et al.'s algorithm looks for a uv-trail in G that consists of (1) a shortest us-path S of G containing 7 guessed vertices and a shortest tv-path T of G containing another 7 guessed vertices such that S and T are anticomplete in G and (2) a shortest st-path Q of $G_{S,T} = G - (N_G[S \cup T] \setminus N_G[\{s,t\}])$. Lemma 5 ensures that much fewer guessed vertices on S and T suffice to guarantee that Q stays intact in $G_{S,T}$. To illustrate the usefulness of Lemma 5, we show in §2 that three lemmas of Berger et al. [7] (i. e., Lemmas 2, 3, and 4) together with Lemma 5 already yield an $O(n^2)$-time uv-trailblazer of degree 4 for G, leading to a simple $O(n^6)$-time trailblazing algorithm on G. More precisely, if a and b (respectively, c and d) are the vertices that are farthest apart from each other in P having the minimum identical distance to u (respectively, v) in G, then (a, b, c, d) is a trail marker for an $O(n^2)$-time uv-trailblazer for G: Due to the symmetry between u and v in G, Lemma 5 guarantees an $O(n^2)$-time obtainable uv-trail of G that contains the precomputed pair of "wings" for this (a, b, c, d).

Our proof of Theorem 1 in §3 further displays the usefulness of Lemma 5. We show that the aforementioned vertices a and b in P actually form a trail marker (a, b) for an $O(n^2 \log^2 n)$-time uv-trailblazer for G. Dropping both c and d from the trail marker (a, b, c, d) of §2 inevitably increases the time of the uv-trailblazer for G. We manage to keep the time of a degree-two uv-trailblazer as low as $O(n^2 \log^2 n)$ via the dynamic data structure of Holm, de Lichtenberg, and Thorup [45] supporting efficient edge updates and connectivity queries for G (see Lemma 8). To make our proof of Theorem 1 in §3 self-contained, a simplified proof of Lemma 4 is included in §2. Since Lemmas 2 and 3 are implied by Lemmas 6 and 7, which are proved in §3, our proof for the $O(n^6)$-time algorithm in §2 is also self-contained.

2 A simple $O(n^6)$-time algorithm

Let G be a connected graph containing vertices u and v. For any vertices x and y of G, let $d_G(x, y) = \|P\|$ for a shortest xy-path P of G. Let $h(x) = d_G(u, x)$ be the *height* of a vertex x in G. If xy is an edge of G, then $|h(x) - h(y)| \leq 1$. For any subgraph H of G, (i) let $G - H = G[V(G) \setminus V(H)]$, (ii) let $N_G(H)$ consist of the vertices $y \in V(G - H)$ adjacent to at least one vertex of H in G, and (iii) let $N_G[H] = N_G(H) \cup V(H)$. For any $x \in V(G)$, let $G - x = G - \{x\}$, let $N_G(x) = N_G(\{x\})$, and let $N_G[x] = N_G[\{x\}]$. X and Y are *adjacent* (respectively, *anticomplete*) in G if $N_G(X) \cap V(Y) \neq \varnothing$ (respectively, $N_G[X] \cap V(Y) = \varnothing$).

▶ **Lemma 2** (Berger et al. [7, Lemma 2.2]). *For any vertices u and v of an n-vertex connected graph G, it takes $O(n^3)$ time to obtain (1) a uv-trail of G or (2) a uv-straight graph H with $V(H) \subseteq V(G)$ such that (a) a uv-trail of G is $O(n^2)$-time obtainable from that of H and (b) if H is uv-trailless, then so is G.*

A path of G is *monotone* [7] if all of its vertices have distinct heights in G. A monotone xy-path of G is a shortest xy-path of G. The converse may not hold. A shortest xy-path of G with $\{x, y\} \cap \{u, v\} \neq \varnothing$ is monotone. A monotone a^*c-path W_1 of G containing a vertex a and a monotone bd^*-path W_2 of G containing a vertex d with

$$h(a^*) + 1 = h(a) = h(b) \leq h(c) = h(d) = h(d^*) - 1$$

form a pair (W_1, W_2) of *wings* for the quadruple (a, b, c, d) of $V(G)$ in G if

$$d_{G[W_1 \cup W_2]}(a^*, d^*) > \|W_1\| + \|W_2\|,$$

that is, $W_1 - c$ (respectively, W_1) and W_2 (respectively, $W_2 - b$) are anticomplete in G. An (a, b, c, d) is *winged* in G if G admits a pair of wings for (a, b, c, d). See also Figure 1 for an example.

▶ **Lemma 3** (Berger et al. [7, Lemma 2.1]). *It takes $O(n^6)$ time to compute a data structure from which the following statements hold for any quadruple (a, b, c, d) of $V(G)$ for an n-vertex graph G:*

1. *It takes $O(1)$ time to determine whether (a, b, c, d) is winged in G.*
2. *If (a, b, c, d) is winged in G, then it takes $O(n)$ time to obtain a pair of wings for (a, b, c, d) in G.*

We comment that Lemma 2.1 of Berger et al. [7] is slightly different from Lemma 3, but their proof is easily adjustable into one for Lemma 3. See also §3 for a proof of Lemma 7, which implies and improves upon Lemma 3.

Let G be a uv-straight graph. If $h(s) - h(t)$ is maximized by the vertices s and t of a uv-path P of G such that $P[u, s]$ is a shortest us-path of G and $P[t, v]$ is a shortest tv-path of G, then the *twist* [7] of P is $h(s) - h(t)$ and we call (s, t) the *twist pair* of P. See also Figure 1 for an example. If (s, t) is the twist pair of a uv-path P of G, then $P[u, s]$ and $P[t, v]$ are disjoint if and only if P is a non-shortest uv-path of G. The next lemma is also needed in §3. To make our proof of Theorem 1 in §3 self-contained, we include a proof of Lemma 4 simplified from that of Berger et al. [7, Lemma 2.3].

▶ **Lemma 4** (Berger et al. [7, Lemma 2.3]). *If (s, t) is the twist pair of a shortest uv-trail P of a uv-straight graph G, then $h(s) \geq h(x) \geq h(t)$ holds for each vertex x of $P[s, t]$.*

Proof. Let $I = V(P[s, t]) \setminus \{s, t\}$. Let s^* (respectively, t^*) be the neighbor of s (respectively, t) in $P[s, t]$. By definition of (s, t), we have $h(s^*) \leq h(s)$ and $h(t^*) \geq h(t)$. If $I = \varnothing$, then $(s^*, t^*) = (t, s)$ implies the lemma. Otherwise, it suffices to prove $h(s) \geq h(x) \geq h(t)$ for each $x \in I$. If $h(x) > h(s)$ were true for the $x \in I$ maximizing the lexicographical order of $(h(x), d_{P[s,t]}(x, t))$, then the concatenation of $P[u, x]$ and a shortest xv-path of G is a uv-trail (containing s^*) of G shorter than P. If $h(x) < h(t)$ were true for the $x \in I$ minimizing the lexicographical order of $(h(x), d_{P[s,t]}(x, t))$, then the concatenation of a shortest ux-path of G and $P[x, v]$ is a uv-trail (containing t^*) of G shorter than P. ◀

A monotone uc-path S of G with $h(c) = h(s)$ is a *sidetrack* for a uv-trail P of G with twist pair (s, t) if satisfying the following *Conditions S*.

S1: $d_{G[S \cup T]}(u, v) > \|S\| + \|T\|$ holds for a monotone tv-path T of G.
S2: The vertex a of S with $h(a) = h(t)$ is on the monotone subpath $P[u, s]$.

The inequality of Condition S1 is equivalent to the statement that $S - c$ (respectively, S) and T (respectively, $T - t$) are anticomplete in G. Thus, $S[a^*, c]$ and $T[t, d^*]$ form a pair of wings for (a, t, c, d) in G, where a^* is the vertex of S with $h(a^*) = h(a) - 1$ and dd^* is the

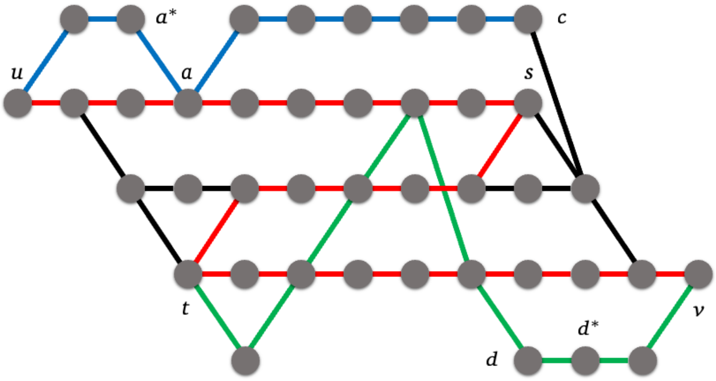


Figure 2 The blue uc-path is a sidetrack S for the red uv-trail P of the uv-straight graph G. Each of $P[t, v]$ and the green tv-path can be a monotone tv-path T satisfying Condition S1.

edge of T with $h(s) = h(d) = h(d^*) - 1$. See Figure 2 for an example. The key to our largely improved uv-trailblazers in §2 and §3 is the following lemma, whose proof is illustrated in Figure 3.

▶ **Lemma 5.** *If S is a sidetrack for a shortest uv-trail P of a uv-straight graph G with twist pair (s, t), then*

$$d_{G[S \cup P[s,t]]}(u, t) \geq d_P(u, t).$$

Proof. Condition S1 implies a monotone tv-path T of G with $d_{G[S \cup T]}(u, v) > \|S\| + \|T\|$. Assume for contradiction a shortest ut-path Q of $G[S \cup P[s, t]]$ with $\|Q\| < d_P(u, t)$, implying $d_{G[Q \cup T]}(u, v) < \|P\|$. By $t \notin V(S)$, Q contains an edge xy with $x \in V(S)$ and $y \in V(P[s, t])$ that minimizes $d_{P[s,t]}(y, t)$. Let R be a shortest uv-path of $G[Q \cup T]$. If x were not in $V(R)$, then $N_G(S[u, x] - x) \cap V(T) \neq \varnothing$, violating Condition S1. Hence, R contains x and, thus, y. Since R is an induced uv-path of G with $\|R\| < \|P\|$, we have $\|R\| = h(v)$, implying that R is monotone. By $d_R(u, x) < d_R(u, y)$,

$$h(x) + 1 = h(y). \tag{1}$$

By $\|Q\| + \|P[t, v]\| < \|P\|$, the concatenation of Q and $P[t, v]$ is a non-induced uv-path of G, implying that $G[Q \cup P[t, v]]$ contains a monotone uv-path R'. Let $x'y'$ be the edge of R' with $x' \in V(S) \cap V(Q)$ and $y' \in V(P[t, v])$ that maximizes $h(y')$. By $d_{R'}(u, x') < d_{R'}(u, y')$,

$$h(x') + 1 = h(y'). \tag{2}$$

We know $h(x') \neq h(t) - 1$ or else $y' = t$ violates Condition S1. We know $h(x') \neq h(t)$ or else Condition S2 violates that P is induced. By $h(x') \geq h(t) + 1$ and Equation (2), y' and t are anticomplete in G. Let t' be the vertex closest to y in $P[y, t]$ with $h(t') = h(t)$, implying that y' and t' are anticomplete in G no matter whether $t' = t$ or not. By $h(x) \geq h(x') \geq h(t) + 1$ and Lemma 4, the concatenation P' of a shortest ut'-path of G, $P[t', y]$, the edge yx, and a shortest xv-path of $G[S[x', x] \cup P[y', v]]$ is an induced uv-path of G shorter than P. By Equation (1) and $d_{P'}(u, x') < d_{P'}(u, y')$, we have that P' is a uv-trail of G, contradicting the definition of P. ◀

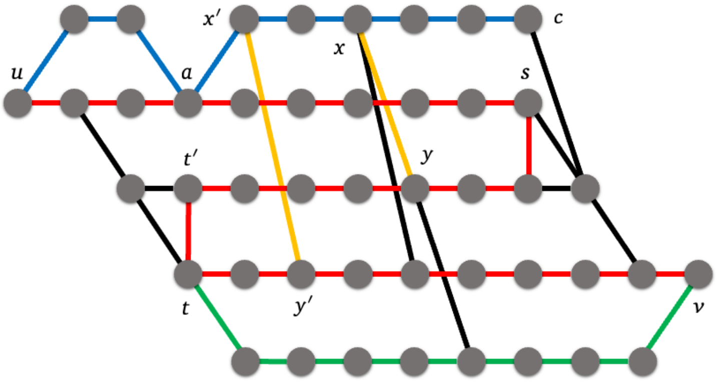


Figure 3 An illustration for the proof of Lemma 5. The red path denotes a shortest uv-trail P of the uv-straight graph G. The blue monotone path denotes a sidetrack S for P. The green path denotes a monotone path T satisfying Condition S1.

We are ready to describe and justify an $O(n^6)$-time algorithm that either reports a uv-trail of G or ensures that G is uv-trailless. Let G be connected without loss of generality.

Our $O(n^6)$-time algorithm

Apply Lemma 2 in $O(n^3)$ time to either report a uv-trail of G as stated in Lemma 2(1) or make G a uv-straight graph satisfying Conditions (a) and (b) of Lemma 2(2) with respect to the original G. If no uv-trail is reported in the previous step, then apply Lemma 3 to obtain the data structure D for the winged quadruples of G in $O(n^6)$ time. With the standard $O(n^2)$-time postprocessing readied by the preprocessing, it remains to show an $O(n^2)$-time degree-4 uv-trailblazer for the uv-straight graph G, which immediately leads to an $O(n^6)$-time trailblazing algorithm that either reports a uv-trail of G or ensures that G is uv-trailless.

Let B be the following $O(n^2)$-time subroutine, taking a quadruple (a, b, c, d) of $V(G)$ as the argument: Determine in $O(1)$ time from the data structure D whether (a, b, c, d) is winged in G. If not, then exit. Otherwise, obtain in $O(n)$ time from D a pair (W_1, W_2) of wings for (a, b, c, d) in G. Since G is uv-straight, it takes $O(n^2)$ time to obtain a monotone uc-path S of G containing W_1 and a monotone bv-path T of G containing W_2. Obtain in $O(n^2)$ time the subgraph $G_{c,b}$ of G induced by

$$\{x \in V(G) : h(b) \leq h(x) \leq h(c)\} \setminus ((N_G[S - c] \cup N_G[T - b]) \setminus \{c, b\}).$$

If c and b are not connected in $G_{c,b}$, then exit. Otherwise, report the concatenation $P_{c,b}$ of (i) the uc-path S, (ii) a shortest cb-path of $G_{c,b}$, and (iii) the bv-path T.

By definition of S, T, and $G_{c,b}$, the uv-path $P_{c,b}$ of G reported by $B(a, b, c, d)$ is induced in G, which is not monotone by $h(b) \leq h(c)$. Thus, $P_{c,b}$ is a uv-trail of G.

Let P be an arbitrary unknown shortest uv-trail of G with twist pair (s, t). Let a (respectively, d) be the vertex of the monotone $P[u, s]$ (respectively, $P[t, v]$) with $h(a) = h(t)$ (respectively, $h(d) = h(s)$). See Figure 4 for an illustration. The rest of the section shows that (a, t, s, d) is a trail marker for B.

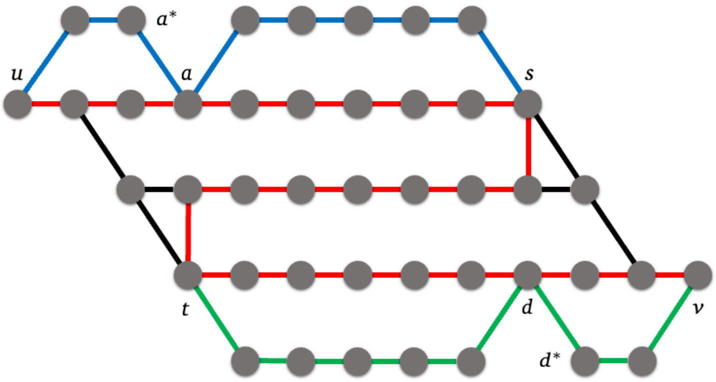


Figure 4 An illustration for the proof that B is a uv-trailblazer of degree four. The red path denotes a shortest uv-trail of the uv-straight graph G. The blue and green paths denote a monotone us-path and a monotone tv-path of G containing a precomputed pair of wings for (a, t, s, d) that need not coincide with P except at a, t, s, and d.

Observe that $P[a^*, s]$ and $P[t, d^*]$ with the neighbor a^* of a in $P[u, a]$ and the neighbor d^* of d in $P[d, v]$ form a pair of wings for (a, t, s, d) in G. Thus, the quadruple (a, t, s, d) is winged in G. The monotone us-path S of G containing W_1 is a sidetrack for P, since the monotone tv-path T of G containing W_2 satisfies Conditions S1 and S2 for S. Due to the symmetry between u and v in G, the monotone vt-path T of the vu-straight graph G is also a sidetrack for the shortest vu-trail P of G with twist pair (t, s), since the monotone su-path S of G satisfies Conditions S1 and S2 for T. Lemma 4 guarantees $h(t) \leq h(x) \leq h(s)$ for each vertex x of $P[s, t]$. By Lemma 5, $P[s, t] - \{s, t\}$ is anticomplete to both $S - s$ and $T - t$, implying that $P[s, t]$ is a path of $G_{s,t}$. Since s and t are connected in $G_{s,t}$, the subroutine call $B(a, t, s, d)$ outputs a uv-trail $P_{s,t}$ of G in $O(n^2)$ time. Hence, (a, t, s, d) is indeed a trail marker of B.

As a matter of fact, $P_{s,t}$ is a shortest uv-trail of G due to $\|P_{s,t}\| = \|P\|$. Since the preprocessing and postprocessing may ruin the shortestness of the reported uv-trail, we have an $O(n^6)$-time algorithm on an n-vertex general (respectively, uv-straight) graph G that either reports a general (respectively, shortest) uv-trail of G or ensures that G is uv-trailless.

3 An $O(n^{4.75})$-time algorithm

This section gives a self-contained proof of Theorem 1. The *product* of $m \times m$ Boolean matrices A and B is the $m \times m$ Boolean matrix C such that $C(i, k) = \mathit{true}$ if and only if $A(i, j) = B(j, k) = \mathit{true}$ holds for an index j. The following lemma implies and improves upon Lemma 2, which takes $O(n^3)$ time to obtain a uv-trail of G from a uv-trail of H.

▶ **Lemma 6.** *For any vertices u and v of an n-vertex connected graph G, it takes $O(M(n))$ time to obtain (1) a uv-trail of G or (2) a uv-straight graph H with $V(H) \subseteq V(G)$ such that (a) a uv-trail of G can be obtained from a uv-trail of H in $O(n^2)$ time and (b) if H is uv-trailless, then so is G.*

Proof. We adopt the proof of Berger et al. [7, Lemma 2.2] with slight simplification and improvement. It takes $O(n^2)$ time to obtain the maximal set $F \subseteq V(G)$ such that $G[F]$ is uv-straight. If $F = V(G)$, then the lemma is proved by returning $H = G$. The rest of the proof assumes $F \subsetneq V(G)$. It takes $O(M(n))$ time to determine whether some connected component K of $G - F$ admits nonadjacent vertices x and y of $N_G(K) \subseteq F$ with $h(x) < h(y)$. If there is such a (K, x, y), then a shortest uv-path of $G[P_x \cup K \cup P_y]$ for any shortest ux-path P_x and yv-path P_y of G is a uv-trail of G obtainable in $O(n^2)$ time, proving the lemma. Otherwise, let H be the union of the uv-straight $G[F]$ and the $O(M(n))$-time obtainable graph H' with $V(H') = F$ (via contracting each connected component of $G - F$ into a single vertex and then squaring the adjacency matrix) such that distinct vertices x and y are adjacent in H' if and only if $\{x, y\} \subseteq N_G(K)$ holds for a connected component K of $G - F$. Observe that each edge xy of H' with $h(x) \neq h(y)$ is also an edge of $G[F]$. By $|h(x)-h(y)| \leq 1$ for all edges xy of H', H remains uv-straight and $d_H(u, x) = h(x)$ holds for each $x \in F$. To see Condition (a), for any given uv-trail Q of H, let P be an $O(n^2)$-time obtainable non-monotone uv-path of G obtained from Q by replacing each edge xy of Q not in $G[F]$ with a shortest xy-path P_{xy} of $G - (F \setminus \{x, y\})$. If P were not induced, then there is an edge zz' of $G[P]$ not in P with $z \in V(P_{xy})$ and $z' \in V(P_{x'y'})$ for distinct edges xy and $x'y'$ of Q that are not in $G[F]$. Thus, $\{x, y, x', y'\} \subseteq N_G(K)$ holds for some connected component K of $G - F$. By definition of H', $H[\{x, y, x', y'\}]$ is complete, contradicting that Q is an induced path of H. Thus, P is a uv-trail of G, proving Condition (a). As for Condition (b), let P be a uv-trail of G. For any distinct vertices x and y of P such that $P[x, y]$ is a maximal subpath of P contained by $G[\{x, y\} \cup K]$ for some connected component K of $G - F$, $P[x, y]$ is an induced xy-path of $G[\{x, y\} \cup K]$. The path Q obtained from P by replacing each such $P[x, y]$ by the edge xy of H' is an induced uv-path of H. If Q were a shortest uv-path of H, then $|h(x) - h(y)| = 1$ holds for each edge xy of Q, implying that each edge xy of Q is an edge of P, contradicting that P is a uv-trail of G. ◀

The bottleneck of our algorithm for Theorem 1 comes from the following lemma, which implies and improves upon Lemma 3 that takes $O(n^6)$ time.

▶ **Lemma 7.** *It takes $\tilde{O}(M(n^2))$ time to compute a data structure from which the following statements hold for any quadruple (a, b, c, d) of $V(G)$ for an n-vertex graph G:*

1. *It takes $O(1)$ time to determine whether (a, b, c, d) is winged in G.*
2. *If (a, b, c, d) is winged in G, then it takes $O(n)$ time to obtain a pair of wings for (a, b, c, d) in G.*

Proof. The lemma holds clearly for the quadruples (a, b, c, d) of $V(G)$ with $h(c) \leq h(a) + 1$. The rest of the proof handles those with $h(a)+2 \leq h(c)$. A pair of wings for such an (a, b, c, d) must be anticomplete in G. It takes $O(n^4)$ time to obtain the $n^2 \times n^2$ Boolean matrix A such that $A((a, b), (c, d)) = \mathit{true}$ if and only if (i) $h(a) = h(b) \leq h(c) = h(d) \leq h(a) + 1$ and (ii) G admits a pair of anticomplete wings for (a, b, c, d). The transitive closure $C = A^n$ of A can be obtained in $O(M(n^2) \cdot \log n)$ time via obtaining A^{2^i} in the i-th iteration. That is, $C((a, b), (c, d)) = \mathit{true}$ if and only if (i) $h(a) = h(b) \leq h(c) = h(d)$ and (ii) G admits a pair of anticomplete wings for (a, b, c, d) in G. Statement 1 is proved. Statement 2 is immediate from the $\tilde{O}(M(n^2))$-time obtainable $n^2 \times n^2$ witness matrix W for C by, e. g., Galil and Margalit [38]. That is, if $C((a, b), (c, d)) = \mathit{true}$ and $h(a) + 2 \leq h(c)$ hold, then $W((a, b), (c, d))$ is a vertex pair (x, y) that satisfies $h(a) < h(x) < h(c)$ and $C((a, b), (x, y)) = C((x, y), (c, d)) = \mathit{true}$. ◀

The following dynamic data structure for a graph supports efficient edge updates and connectivity queries.

▶ **Lemma 8** (Holm, de Lichtenberg, and Thorup [45]). *There is a data structure for an initially empty n-vertex graph that supports each edge insertion and edge deletion in amortized $O(\log^2 n)$ time and answers whether two vertices are connected in $O(\log n/ \log\log n)$ time.*

We are ready to prove Theorem 1. Assume without loss of generality that G is connected.

Our $O(n^{4.75})$-time algorithm

Apply Lemma 6 in $O(M(n))$ time to either report a uv-trail of G as in Lemma 6(1) or make G a uv-straight graph satisfying Conditions (a) and (b) of Lemma 6(2) with respect to the original G. If no uv-trail is reported in the previous step, then apply Lemma 7 in $\tilde{O}(M(n^2))$ time to obtain the data structure D for the winged quadruples of $V(G)$ in G. It remains to show an $O(n^2 \log^2 n)$-time degree-two uv-trailblazer for the uv-straight graph G based on the precomputed D which already spends $O(n^{4.75})$ time. We proceed in two phases. Phase 1 shows that we already have an $O(n^3)$-time degree-two uv-trailblazer for G. Phase 2 then reduces the time to $O(n^2 \log^2 n)$ via Lemma 8.

Phase 1. Let B_1 be the $O(n^3)$-time subroutine, taking a pair (a, b) of $V(G)$ as the only argument, that runs the following $O(n^2)$-time procedure for each vertex c of G: Determine from D in $O(n)$ time whether G admits a winged quadruple (a, b, c, d_c) of $V(G)$ for some d_c. If not, then exit. Otherwise, obtain from D in $O(n)$ time a pair (W_1, W_2) of wings for an arbitrary winged (a, b, c, d_c). Since G is uv-straight, it takes $O(n^2)$ time to obtain a monotone uc-path S_c of G containing W_1 and a monotone bv-path T_c of G containing W_2. Obtain in $O(n^2)$ time the subgraph G_c of G induced by

$$(\{x \in V(G) : h(b) \leq h(x) \leq h(c)\} \setminus (N_G[S_c - c] \setminus \{c\})) \cup V(T_c).$$

If the vertices c and b are not connected in G_c, then exit. Otherwise, report the $O(n^2)$-time obtainable concatenation P_c of the uc-path S_c of G and a shortest cv-path of G_c.

By definition of S_c, T_c, and G_c, the uv-path P_c of G reported by $B_1(a, b)$ for any c is induced in G. Since the height of each neighbor of c in G_c is at most $h(c)$, P_c is not monotone. Thus, P_c is a uv-trail of G. Let P be an arbitrary unknown shortest uv-trail of G with twist pair (s, t). Let a (respectively, e) be the vertex of the monotone $P[u, s]$ (respectively, $P[t, v]$) with $h(a) = h(t)$ (respectively, $h(e) = h(s)$). See Figure 5 for an illustration. To ensure that B_1 is an $O(n^3)$-time uv-trailblazer of degree 2 for G, the rest of the phase proves that (a, t) is a trail marker for B_1 by showing that the iteration with $c = s$ reports a uv-trail P_s of G.

Let a^* be the neighbor of a in the monotone $P[u, a]$, implying $h(a^*) = h(t) - 1$. Let e^* be the neighbor of e in the monotone $P[e, v]$, implying $h(e^*) = h(s) + 1$. Since $P[a^*, s]$ and $P[t, e^*]$ form a pair of wings for (a, t, s, e) in G, there is a d_s such that (a, t, s, d_s) is winged in G. Let (W_1, W_2) be the pair of wings for (a, t, s, d_s) in G obtained from D. The monotone us-path S_s of G containing W_1 is a sidetrack for P, since the monotone tv-path T_s of G containing W_2 satisfies Conditions S1 and S2 for S_s. By Lemma 4, each vertex x of $P[s, t]$ satisfies $h(t) \leq h(x) \leq h(s)$. By Lemma 5, $S_s - s$ and $P[s, t] - s$ are anticomplete in G, implying that $P[s, t]$ is a path of G_s. Since s and t are connected in G_s, the subroutine call $B_1(a, t)$ outputs a uv-trail P_s of G in the iteration with $c = s$. Hence, (a, t) is indeed a trail marker of B. One can verify that P_s is also a shortest uv-trail of the uv-straight G, although d_s need not be e. Thus, we have an $O(n^5)$-time algorithm on an n-vertex general (respectively, uv-straight) graph G that either reports a general (respectively, shortest) uv-trail of G or ensures that G is uv-trailless.

Phase 2. Since many prefixes of a long sidetrack for a shortest uv-trail P of G remain sidetracks for P, an edge can be deleted and then inserted back $\Omega(n)$ times in Phase 1. Phase 2 avoids the redundancy by processing the sidetracks in the decreasing order of their

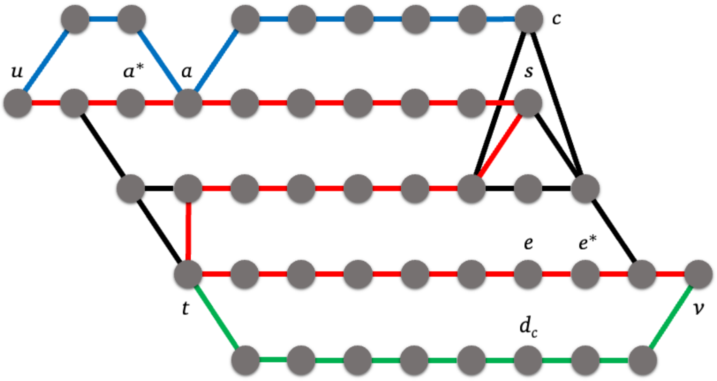


Figure 5 An illustration for the proof that B_1 is a uv-trailblazer of degree two. The red path denotes a shortest uv-trail P of the uv-straight graph G. The blue and green paths denote a monotone uc-path S_c and a monotone tv-path T_c of G containing a precomputed pair of wings for (a, t, c, d_c) that need not coincide with P except at a and t.

lengths. Let B_2 be the following subroutine that takes a pair (a, b) of $V(G)$ as the only argument. Obtain in overall $O(n^2)$ time from D each set C_i with $0 \le i \le h(v)$ that consists of the vertices c of G with $h(c) = i$ such that G admits a winged quadruple (a, b, c, d_c) for some vertex d_c. Let C be the union of all C_i with $0 \le i \le h(v)$. Obtain in overall $O(n^2)$ time from D for each vertex $c \in C$ (i) a monotone uc-path S_c of G containing a and (ii) a monotone bv-path T_c with

$$d_{G[S_c \cup T_c]}(u, v) > \|S_c\| + \|T_c\|.$$

Obtain the subgraph H of G induced by the vertices with heights at least $h(a)$ in $O(n^2 \log^2 n)$ time by the dynamic data structure of Lemma 8. Iteratively perform the following steps in the decreasing order of the indices i with $h(a) \le i < h(v)$:

1. Delete from H the incident edges of $N_G[S_c - c]$ in G for all $c \in C_i$.
2. Insert to H the incident edges of C_i in G.
3. Delete from H all edges xy of G with $h(x) = i$ and $h(y) = i + 1$.
4. If b is not connected to any $c \in C_i$ in H, then proceed to the next iteration. Otherwise, let c be an arbitrary vertex of C_i that is connected to b in H. Exit the loop and report the $O(n^2)$-time obtainable concatenation P_c of S_c and a shortest cv-path of $G[H \cup T_c]$.

Since $S_c - c$ and $T_c - b$ are anticomplete in G and the height of each neighbor of c in H is at most $h(c)$, any arbitrary reported uv-path P_c of G is a uv-trail of G.

Throughout all iterations, the incident edges of each vertex of G is deleted $O(1)$ times by Step 1, each edge of G is updated $O(1)$ times by Steps 2 and 3, and each vertex $c \in C$ is queried $O(1)$ times by Step 4. Thus, each subroutine call $B_2(a, b)$ runs in $O(n^2 \log^2 n)$ time.

Let P be an arbitrary unknown shortest uv-trail of G with twist pair (s, t). As in Phase 1, let a (respectively, e) be the vertex of the monotone $P[u, s]$ (respectively, $P[t, v]$) with $h(a) = h(t)$ (respectively, $h(e) = h(s)$). The rest of the phase proves that (a, t) is a trail marker for B_2 by showing that an iteration with $i \ge h(s)$ in the loop of the subroutine call $B_2(a, t)$ reports a uv-trail P_c of G. See Figure 6 for an illustration.

If an iteration of $B_2(a, t)$ with $i \ge h(s) + 1$ reports a uv-trail of G (that need not be shortest), then we are done. Otherwise, we show that the iteration with $i = h(s)$ has to report a uv-trail of G. For each $c \in C$ with $h(c) \ge i$, let s_c be the unknown vertex of S_c

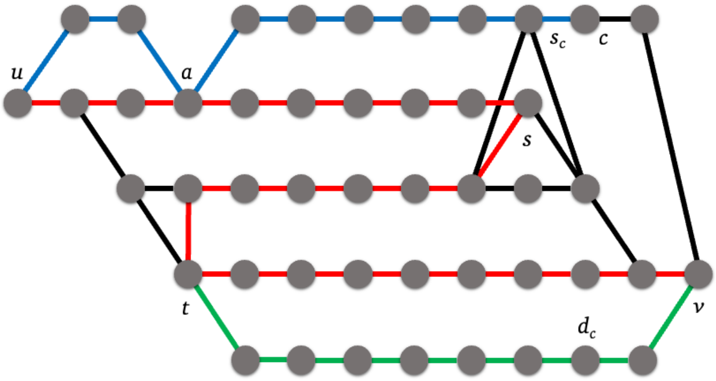


Figure 6 An illustration for the proof that B_2 is a uv-trailblazer of degree two. The red path denotes a shortest uv-trail P of the uv-straight graph G. The blue and green paths denote a monotone uc-path S_c and a monotone tv-path T_c of G containing a precomputed pair of wings for (a, t, c, d_c) that need not coincide with P except at a and t. $S_c[u, s_c]$ remains a sidetrack for P.

with $h(s_c) = i$. $S_c[u, s_c]$ remains a sidetrack for P, since T_c still satisfies Conditions S1 and S2 for $S_c[u, s_c]$. Thus, $s_c \in C_i$. By Lemma 5, $S_c[u, s_c] - s_c$ and $P[s, t] - s$ are anticomplete in G even if $S_c[u, s_c]$ need not be S_{s_c}. As a result, $P[s, t] - s$ is a path of the H at the completion of Step 1 in the i-th iteration. By $s \in C_i$ and Lemma 4, $P[s, t]$ is a path of the graph H at the completion of Step 3 in the i-th iteration. Therefore, s is a $c \in C_i$ that is connected to t in H. Step 4 in the i-th iteration has to output a (shortest) uv-trail P_c of G for some $c \in C_i$ that need not be s. Thus, we have an $O(n^{4.75})$-time algorithm that either obtains a uv-trail of G or ensures that G is uv-trailless. A reported uv-trail of G by this $O(n^{4.75})$-time algorithm need not be a shortest uv-trail of G, since we cannot afford to spend $O(n^2)$ time, as in Phase 1, for each $c \in C$ that is connected to t in the H at the $h(c)$-th iteration to obtain a shortest cv-path of $G[H \cup T_c]$.

4 Concluding remarks

We show an $O(n^{4.75})$-time algorithm for computing a uv-trail of an n-vertex undirected unweighted graph G with $\{u, v\} \subseteq V(G)$. The key to our improved algorithm is the observation regarding an arbitrary shortest uv-trail of a uv-straight graph G described by Lemma 5. The inequality of Lemma 5 is stronger than the condition that $S - c$ and $P[s, t] - s$ are anticomplete in G. As a matter of fact, the latter suffices for our uv-trailblazers in §2 and §3. Thus, a further improved uv-trailblazer might be possible if the wings for a winged quadruple can be obtained more efficiently. As mentioned in Phase 1 of §3, a shortest uv-trail, if any, of a uv-straight G can be obtained by our B_1-based trailblazing algorithm in $O(n^5)$ time. Detecting a uv-trail with length at least $2d_G(u, v)$ is NP-complete [7, Theorem 1.6]. It would be of interest to see if a shortest uv-trail or a uv-trail having length at least $d_G(u, v)+k$ for a positive $k = O(1)$ in a general G can be obtained in polynomial time. It is also of interest to see whether the one-to-all (respectively, all-pairs) version of the problem can be solved in time much lower than $O(n^{5.75})$ (respectively, $O(n^{6.75})$).

References

1 Tara Abrishami, Maria Chudnovsky, Marcin Pilipczuk, Pawel Rzazewski, and Paul D. Seymour. Induced subgraphs of bounded treewidth and the container method. In Dániel Marx, editor, *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1948–1964, 2021. `doi:10.1137/1.9781611976465.116`.

2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 522–539, 2021. `doi:10.1137/1.9781611976465.32`.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. `doi:10.1145/210332.210337`.

4 Claude Berge. Les problèmes de coloration en théorie des graphes. *Publications de l'Institut de statistique de l'Université de Paris*, 9:123–160, 1960.

5 Claude Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung). *Wissenschaftliche Zeitschrift, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe*, 10:114–115, 1961.

6 Claude Berge. *Graphs*. North-Holland, Amsterdam, New York, 1985.

7 Eli Berger, Paul D. Seymour, and Sophie Spirkl. Finding an induced path that is not a shortest path. *Discrete Mathematics*, 344(7):112398.1–112398.6, 2021. `doi:10.1016/j.disc.2021.112398`.

8 Daniel Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991. See [9] for corrigendum. `doi:10.1016/0012-365X(91)90098-M`.

9 Daniel Bienstock. Corrigendum to: D. Bienstock, "On the complexity of testing for odd holes and induced odd paths" Discrete Mathematics 90 (1991) 85–92. *Discrete Mathematics*, 102(1):109, 1992. `doi:10.1016/0012-365X(92)90357-L`.

10 Hsien-Chih Chang and Hsueh-I Lu. A faster algorithm to recognize even-hole-free graphs. *Journal of Combinatorial Theory, Series B*, 113:141–161, 2015. `doi:10.1016/j.jctb.2015.02.001`.

11 Yijia Chen and Jörg Flum. On parameterized path and chordless path problems. In *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity*, pages 250–263, 2007. `doi:10.1109/CCC.2007.21`.

12 Hou-Teng Cheong and Hsueh-I Lu. Finding a shortest even hole in polynomial time. *Journal of Graph Theory*, 2022, to appear. `doi:10.1002/jgt.22748`.

13 Maria Chudnovsky, Gérard Cornuéjols, Xinming Liu, Paul D. Seymour, and Kristina Vušković. Recognizing Berge graphs. *Combinatorica*, 25(2):143–186, 2005. `doi:10.1007/s00493-005-0012-8`.

14 Maria Chudnovsky, Ken-ichi Kawarabayashi, and Paul Seymour. Detecting even holes. *Journal of Graph Theory*, 48(2):85–111, 2005. `doi:10.1002/jgt.20040`.

15 Maria Chudnovsky, Marcin Pilipczuk, Michal Pilipczuk, and Stéphan Thomassé. On the maximum weight independent set problem in graphs without induced cycles of length at least five. *SIAM Journal on Discrete Mathematics*, 34(2):1472–1483, 2020. `doi:10.1137/19M1249473`.

16 Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006. `doi:10.4007/annals.2006.164.51`.

17 Maria Chudnovsky, Alex Scott, and Paul Seymour. Detecting a long odd hole. *Combinatorica*, 41(1):1–30, 2021. `doi:10.1007/s00493-020-4301-z`.

18 Maria Chudnovsky, Alex Scott, and Paul Seymour. Finding a shortest odd hole. *ACM Transactions on Algorithms*, 17(2):13.1–13.21, 2021. `doi:10.1145/3447869`.

19 Maria Chudnovsky, Alex Scott, Paul Seymour, and Sophie Spirkl. Detecting an odd hole. *Journal of the ACM*, 67(1):5.1–5.12, 2020. `doi:10.1145/3375720`.

20 Maria Chudnovsky and Paul Seymour. The three-in-a-tree problem. *Combinatorica*, 30(4):387–417, 2010. doi:10.1007/s00493-010-2334-4.

21 Maria Chudnovsky and Paul D. Seymour. Even pairs in Berge graphs. *Journal of Combinatorial Theory, Series B*, 99(2):370–377, 2009. doi:10.1016/j.jctb.2008.08.002.

22 Maria Chudnovsky and Vaidy Sivaraman. Odd holes in bull-free graphs. *SIAM Journal on Discrete Mathematics*, 32(2):951–955, 2018. doi:10.1137/17M1131301.

23 Michele Conforti, Gérard Cornuéjols, Ajai Kapoor, and Kristina Vušković. Finding an even hole in a graph. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 480–485, 1997. doi:10.1109/SFCS.1997.646136.

24 Michele Conforti, Gérard Cornuéjols, Ajai Kapoor, and Kristina Vušković. Even and odd holes in cap-free graphs. *Journal of Graph Theory*, 30(4):289–308, 1999. doi:10.1002/(SICI)1097-0118(199904)30:4\%3C289::AID-JGT4\%3E3.0.CO;2-3.

25 Michele Conforti, Gérard Cornuéjols, Ajai Kapoor, and Kristina Vušković. Even-hole-free graphs Part I: Decomposition theorem. *Journal of Graph Theory*, 39(1):6–49, 2002. doi:10.1002/jgt.10006.

26 Michele Conforti, Gérard Cornuéjols, Ajai Kapoor, and Kristina Vušković. Even-hole-free graphs Part II: Recognition algorithm. *Journal of Graph Theory*, 40(4):238–266, 2002. doi:10.1002/jgt.10045.

27 Michele Conforti, Gérard Cornuéjols, Xinming Liu, Kristina Vušković, and Giacomo Zambelli. Odd hole recognition in graphs of bounded clique size. *SIAM Journal on Discrete Mathematics*, 20(1):42–48, 2006. doi:10.1137/S089548010444540X.

28 Linda Cook, Jake Horsfield, Myriam Preissmann, Cléophée Robin, Paul Seymour, Ni Luh Dewi Sintiari, Nicolas Trotignon, and Kristina Vušković. Graphs with all holes the same length. *arXiv*, 2021. arXiv:2110.09970.

29 Linda Cook and Paul Seymour. Detecting a long even hole. *arXiv*, 2020. arXiv:2009.05691.

30 Gérard Cornuéjols, Xinming Liu, and Kristina Vušković. A polynomial algorithm for recognizing perfect graphs. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 20–27, 2003. doi:10.1109/SFCS.2003.1238177.

31 Murilo Vicente Gonçalves da Silva and Kristina Vušković. Decomposition of even-hole-free graphs with star cutsets and 2-joins. *Journal of Combinatorial Theory, Series B*, 103(1):144–183, 2013. doi:10.1016/j.jctb.2012.10.001.

32 Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. Finding even cycles faster via capped k-walks. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM Symposium on Theory of Computing*, pages 112–120. ACM, 2017. doi:10.1145/3055399.3055459.

33 Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: hardness for all induced patterns and faster non-induced cycles. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, pages 1167–1178, 2019. doi:10.1145/3313276.3316329.

34 David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998. doi:10.1137/S0097539795290477.

35 Hazel Everett, Celina M. H. de Figueiredo, Cláudia Linhares Sales, Frédéric Maffray, Oscar Porto, and Bruce A. Reed. Path parity and perfection. *Discrete Mathematics*, 165-166:233–252, 1997. doi:10.1016/S0012-365X(96)00174-4.

36 Michael R. Fellows, Jan Kratochvíl, Matthias Middendorf, and Frank Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13(3):266–282, 1995. doi:10.1007/BF01190507.

37 J. Fonlupt and J.P. Uhry. Transformations which preserve perfectness and H-perfectness of graphs. In Achim Bachem, Martin Grötschel, and Bemhard Korte, editors, *Bonn Workshop on Combinatorial Optimization*, volume 66 of *North-Holland Mathematics Studies*, pages 83–95. North-Holland, 1982. doi:10.1016/S0304-0208(08)72445-9.

38 Zvi Galil and Oded Margalit. Witnesses for boolean matrix multiplication and for transitive closure. *Journal of Complexity*, 9(2):201–221, 1993. doi:10.1006/jcom.1993.1014.

39 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

40 Serge Gaspers, Shenwei Huang, and Daniël Paulusma. Colouring square-free graphs without long induced paths. In Rolf Niedermeier and Brigitte Vallée, editors, *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science*, LIPIcs 96, pages 35.1–35.15, 2018. doi:10.4230/LIPIcs.STACS.2018.35.

41 Emilio Di Giacomo, Giuseppe Liotta, and Tamara Mchedlidze. Lower and upper bounds for long induced paths in 3-connected planar graphs. *Theoretical Computer Science*, 636:47–55, 2016. doi:10.1016/j.tcs.2016.04.034.

42 Petr A. Golovach, Daniël Paulusma, and Erik Jan van Leeuwen. Induced disjoint paths in AT-free graphs. In Fedor V. Fomin and Petteri Kaski, editors, *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory*, Lecture Notes in Computer Science 7357, pages 153–164, 2012. doi:10.1007/978-3-642-31155-0_14.

43 Robert Haas and Michael Hoffmann. Chordless paths through three vertices. *Theoretical Computer Science*, 351(3):360–371, 2006. doi:10.1016/j.tcs.2005.10.021.

44 Chính T. Hoàng, Marcin Kaminski, Joe Sawada, and R. Sritharan. Finding and listing induced paths and cycles. *Discrete Applied Mathematics*, 161(4-5):633–641, 2013. doi:10.1016/j.dam.2012.01.024.

45 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, July 2001. doi:10.1145/502090.502095.

46 Wen-Lian Hsu. Recognizing planar perfect graphs. *Journal of the ACM*, 34(2):255–288, 1987. doi:10.1145/23005.31330.

47 Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Mim-width I. induced path problems. *Discrete Applied Mathematics*, 278:153–168, 2020. doi:10.1016/j.dam.2019.06.026.

48 David S. Johnson. The NP-completeness column. *ACM Transactions on Algorithms*, 1(1):160–176, 2005. doi:10.1145/1077464.1077476.

49 Marcin Kaminski and Naomi Nishimura. Finding an induced path of given parity in planar graphs in polynomial time. In Yuval Rabani, editor, *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 656–670, 2012. doi:10.1137/1.9781611973099.55.

50 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

51 Matthias Kriesell. Induced paths in 5-connected graphs. *Journal of Graph Theory*, 36(1):52–58, 2001. doi:10.1002/1097-0118(200101)36:1\%3C52::AID-JGT5\%3E3.0.CO;2-N.

52 Kai-Yuan Lai, Hsueh-I Lu, and Mikkel Thorup. Three-in-a-tree in near linear time. In *Proccedings of the 52nd Annual ACM Symposium on Theory of Computing*, pages 1279–1292, 2020. doi:10.1145/3357713.3384235.

53 François Le Gall. Powers of tensors and fast matrix multiplication. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

54 Wei Liu and Nicolas Trotignon. The k-in-a-tree problem for graphs of girth at least k. *Discrete Applied Mathematics*, 158(15):1644–1649, 2010. doi:10.1016/j.dam.2010.06.005.

55 Henry Meyniel. A new property of critical imperfect graphs and some consequences. *European Journal of Combinatorics*, 8(3):313–316, 1987. doi:10.1016/S0195-6698(87)80037-9.

56 Oscar Porto. Even induced cycles in planar graphs. In *Proceedings of the 1st Latin American Symposium on Theoretical Informatics*, pages 417–429, 1992. doi:10.1007/BFb0023845.

57 Marko Radovanovic, Nicolas Trotignon, and Kristina Vušković. The (theta, wheel)-free graphs part IV: induced paths and cycles. *Journal of Combinatorial Theory, Series B*, 146:495–531, 2021. doi:10.1016/j.jctb.2020.06.002.

58 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

59 D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi:10.1137/0205021.

60 Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984. see [61] for addendum. doi:10.1137/0213035.

61 Robert Endre Tarjan and Mihalis Yannakakis. Addendum: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 14(1):254–255, 1985. doi:10.1137/0214020.

62 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith–Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

63 Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997. doi:10.1137/S0895480194274133.

Depth Lower Bounds in Stabbing Planes for Combinatorial Principles

Stefan Dantchev ✉
Department of Computer Science, Durham University, UK

Nicola Galesi ✉
Department of Computer Science, Sapienza University of Rome, Italy

Abdul Ghani ✉
Department of Computer Science, Durham University, UK

Barnaby Martin ✉
Department of Computer Science, Durham University, UK

Abstract

Stabbing Planes is a proof system introduced very recently which, informally speaking, extends the DPLL method by branching on integer linear inequalities instead of single variables. The techniques known so far to prove size and depth lower bounds for Stabbing Planes are generalizations of those used for the *Cutting Planes* proof system established via communication complexity arguments. Rank lower bounds for Cutting Planes are also obtained by geometric arguments called *protection lemmas*.

In this work we introduce two new geometric approaches to prove size/depth lower bounds in Stabbing Planes working for any formula: (1) the *antichain method*, relying on *Sperner's Theorem* and (2) the *covering method* which uses results on *essential coverings* of the boolean cube by linear polynomials, which in turn relies on *Alon's combinatorial Nullenstellensatz*.

We demonstrate their use on classes of combinatorial principles such as the *Pigeonhole principle*, the *Tseitin contradictions* and the *Linear Ordering Principle*. By the first method we prove almost linear size lower bounds and optimal logarithmic depth lower bounds for the Pigeonhole principle and analogous lower bounds for the Tseitin contradictions over the complete graph and for the Linear Ordering Principle. By the covering method we obtain a superlinear size lower bound and a logarithmic depth lower bound for Stabbing Planes proof of Tseitin contradictions over a grid graph.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Proof complexity

Keywords and phrases proof complexity, computational complexity, lower bounds, cutting planes, stabbing planes

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.24

Related Version *Previous Version*: https://arxiv.org/abs/2102.07622

Acknowledgements While finishing the writing of this manuscript we learned about [12] from Noah Fleming. We would like to thank him for answering some questions on his paper [2], and sending us the manuscript [12] and for comments on a preliminary version of this work.

1 Introduction

Finding a satisfying assignment for a propositional formula (SAT) is a central component for many computationally hard problems. Despite being older than 50 years and exponential time in the worst-case, the DPLL algorithm [10, 11, 26] is the core of essentially all high performance modern SAT-solvers. DPLL is a recursive boolean method: at each call one variable x of the formula $\mathcal{F}$ is chosen and the search recursively branches into the two cases obtained by setting x respectively to 1 and 0 in $\mathcal{F}$. It is well-known that the execution trace of the DPLL algorithm running on an unsatisfiable formula $\mathcal{F}$ is nothing more than a treelike refutation of $\mathcal{F}$ in the proof system of *Resolution* [26] (Res).

Since SAT can be viewed as an optimization problem the question whether Integer Linear Programming (ILP) can be made feasible for satisfiability testing received a lot of attention and is considered among the most challenging problems in local search [27, 17]. One proof system capturing ILP approaches to SAT is *Cutting Planes*, a system whose main rule implements the *rounding* (or *Chvátal cut*) approach to ILP. Cutting planes works with integer linear inequalities of the form $\mathbf{ax} \leq b$, with $\mathbf{a}, b$ integers, and, like resolution, is a sound and complete refutational proof system for CNF formulas: indeed a clause $C = (x_1 \vee \ldots \vee x_r \vee \neg y_1 \vee \ldots \vee \neg y_s)$ can be written as the integer inequality $y_1 + \cdots + y_s - x_1 - \cdots - x_r \leq s - 1$.

Beame et al. [2], extended the idea of DPLL to a more general proof strategy based on ILP. Instead of branching only on a variable as in DPLL, in this method one considers a pair $(\mathbf{a}, b)$, with $\mathbf{a} \in \mathbb{Z}^n$ and $b \in \mathbb{Z}$, and branches limiting the search to the two half-planes: $\mathbf{ax} \leq b - 1$ and $\mathbf{ax} \geq b$. A *path* terminates when the LP defined by the inequalities in $\mathcal{F}$ and those forming the path is infeasible. This method can be made into a refutational treelike proof system for UNSAT CNF's called *Stabbing Planes* (SP) ([2]) and it turned out that it is polynomially equivalent to the treelike version of Res(CP), a proof system introduced by Krajíček [19] where clauses are disjunction of linear inequalities.

In this work we consider the complexity of proofs in SP focusing on the *length*, i.e. the number of queries in the proof; the *depth* (called also *rank* in [2]), i.e. the length of the longest path in the proof tree; and the *size*, i.e. the bit size of all the coefficients appearing in the proof.

1.1 Previous works and motivations

After its introduction as a proof system in the work [2] by Beame, Fleming, Impagliazzo, Kolokolova, Pankratov, Pitassi and Robere, *Stabbing Planes* received great attention. The quasipolynomial upper bound for the size of refuting Tseitin contradictions in SP given in [2] was surprisingly extended to CP in the work of [9] of Dadush and Tiwari refuting a long-standing conjecture. Recently in [12], Fleming, Göös, Impagliazzo, Pitassi, Robere, Tan and Wigderson were further developing the initial results proved in [2] making important progress on the question whether all Stabbing Planes proofs can be somehow efficiently simulated by Cutting Planes showing quasipolynomial simulation of bounded weight SP by CP.

Significant lower bounds for size can be obtained in SP, but in a limited way, using modern developments of a technique for CP based on communication complexity of search problems introduced by Impagliazzo, Pitassi, Urquhart in [16]: in [2] it is proven that size S and depth D SP refutations imply treelike Res(CP) proofs of size $O(S)$ and width $O(D)$; Kojevnikov [18], improving the *interpolation method* introduced for Res(CP) by Krajíček [19], gave exponential lower bounds for treelike Res(CP) when the width of the clauses (i.e. the number of linear inequalities in a clause) is bounded by $o(n/\log n)$. Hence these lower bounds are applicable only to very specific classes of formulas (whose hardness comes from boolean circuit hardness) and only to SP refutations of low depth.

Nevertheless SP appears to be a strong proof system. Firstly notice that the condition terminating a path in a proof is not a trivial contradiction like in resolution, but is the infeasibility of an LP, which is only a polynomial time verifiable condition. Hence linear size SP proofs might be already a strong class of SP proofs, since they can hide a polynomial growth into one final node whence to run the verification of the terminating condition.

Rank and depth in CP and SP

It is known that, contrary to the case of other proof systems like Frege, neither CP nor SP proofs can be balanced (see [2]), in the sense that a size $2^{O(d)}$ proof can always be converted into a depth $O(d)$ proof. The depth of CP-proofs of a set of linear inequalities L is measured in two ways: (1) as the depth of the dag representing to the proof, and (2) by the *Chvátal rank* of the associated polytope P.[1] It is known that rank in CP and depth in SP are separated, in the sense that Tseitin formulas can be proved in depth $O(\log n)$ in SP [2], but are known to require rank $\Theta(n)$ to be refuted in CP [6]. In this paper we further develop the study of proof depth for SP.

Rank lower bound techniques for Cutting Planes are essentially of two types. The main method is by reducing to the *real communication complexity* of certain search problem [16]. As such this method mainly works for classes of formulas *lifted* by certain gadgets capturing specific boolean functions. A second class of methods have been developed for Cutting Planes, which lower bound the rank measures of a polytope. In this setting, lower bounds are typically proven using a geometric method called *protection lemmas* [6]. These methods were recently extended in [12] also to the case of Semantic Cutting Planes. In principle this geometric method can be applied to any formula and not only to the lifted ones, furthermore for many formulas (such as the Tseitin formulas) it is known how to achieve $\Omega(n)$ rank lower bounds in CP via protection lemmas, while proving even $\omega(\log n)$ lower bounds via real communication complexity is impossible, due to a known folklore upper bound.

Lower bounds for depth in Stabbing Planes, proved in [2], are instead obtained only as a consequence of the real communication complexity approach extended to Stabbing Planes. In this paper we introduce two geometric approaches to prove depth lower bounds in SP.

Specifically the results we know at present relating SP and CP are:

1. SP polynomially simulates CP (Theorem 4.5 in [2]) and CP polynomially simulates SP with bounded coefficients [12]. Hence in particular the PHP_n^m can be refuted in SP by a proof of size $O(n^2)$ ([8]). Furthermore it can be refuted by a $O(\log n)$ depth proof since polynomial size CP proofs, by Theorem 4.4 in [2], can be balanced in SP.[2]
2. Beame et al. in [2] proved the surprising result that the class of Tseitin contradictions $\mathsf{Ts}(G, \omega)$ over any graph G of maximum degree D, with an odd charging ω, can be refuted in SP in size quasipolynomial in $|G|$ and depth $O(\log^2 |G| + D)$.

Depth lower bounds for SP are proved in [2]:

1. A $\Omega(n/\log^2 n)$ lower bound for the formula $\mathsf{Ts}(G, w) \circ \mathsf{VER}^n$, composing $\mathsf{Ts}(G, \omega)$ (over an expander graph G) with the gadget function VER^n (see Theorem 5.7 in [2] for details).
2. A $\Omega(\sqrt{n \log n})$ lower bound for the formula $\mathsf{Peb}(G) \circ \mathsf{IND}_l^n$ over $n^5 + n \log n$ variables obtained by lifting a pebbling formula $\mathsf{Peb}(G)$ over a graph with high pebbling number, with a *pointer function* gadget IND_l^n (see Theorem 5.5. in [2] for details).
3. There are also sub-linear lower bounds on SP depth when the coefficients in the SP proof are of magnitude at most 2^{n^δ} for some constant δ for random $O(\log n)$-CNF formulas over n variables. This lower bound is hence obtained (by communication complexity arguments) for an unlifted class of formulas by combining the Cutting Planes size lower bounds for

[1] This is the minimal d such that $P^{(d)}$ is empty, where $P^{(0)}$ is the polytope associated to L and $P^{(i+1)}$ is the polytope defined by all inequalities which can be inferred from those in $P^{(i)}$ using one Chvátal cut.

[2] Another way of proving this result is using Theorem 4.8 in [2] stating that if there are length L and space S CP refutations of a set of linear integral inequalities, then there are depth $O(S \log L)$ SP refutations of the same set of linear integral inequalities; and then use the result in [14] (Theorem 5.1) that PHP_n^m has polynomial length and constant space CP refutations.

random $O(\log n)$-CNF formulas of [15, 13] with the quasipolynomial transformation of Stabbing Planes proofs into Cutting Planes for bounded coefficients (2^{n^δ}, see [12]). This gives a $\exp(n^\delta/\,\mathsf{polylog}(n))$ size lower bound, and thus a $\Omega(n^\delta/\,\mathsf{polylog}(n))$ depth lower bound for SP proofs.

Similar to size, these depth lower bounds are applicable only to very specific classes of formulas. In fact they are obtained by extending to SP the technique introduced in [16, 20] for CP of reducing shallow proofs of a formula $\mathcal{F}$ to efficient real communication protocols computing a related search problem and then proving that such efficient protocols cannot exist.

Despite the fact that SP is at least as strong as CP, in SP the known lower bound techniques are derived from those of CP. Hence finding other techniques to prove depth and size lower bounds for SP is important to understand its proof strength. For instance, unlike CP where we know tight $\Theta(\log n)$ rank bounds for the PHP_n^m [6, 25] and $\Omega(n)$ rank bounds for Tseitin contradictions [6], for SP no depth lower bound is at present known for purely combinatorial statements.

In this work we address such problems.

1.2 Contributions and techniques

The main motivation of this work was to study size and depth lower bounds in SP through new methods, possibly geometric. Differently from weaker systems like Resolution, except for the technique highlighted above and based on reducing to the communication complexity of search problems, we do not know of other methods to prove size and depth lower bounds in SP. In CP and Semantic CP instead geometrical methods based on protection lemmas were used to prove rank lower bounds in [6, 12].

Our first steps in this direction were to set up methods working for truly combinatorial statements, like $\mathsf{Ts}(G, w)$ or PHP_n^m, which we know to be efficiently provable in SP, but on which we cannot use methods reducing to the complexity of boolean functions, like the ones based on communication complexity.

We present two new methods for proving depth lower bounds in SP which in fact are the consequence of proving length lower bounds that do not depend on the bit-size of the coefficients.

As applications of our two methods we respectively prove:

1. An exponential separation between the rank in CP and the depth in SP, using a new counting principle which we introduce and that we call the *Simple Pigeonhole Principle* SPHP_n. We prove that SPHP_n has $O(1)$ rank in CP and requires $\Omega(\log n)$ depth in SP. Together with the results proving that Tseitin formulas requires $\Omega(n)$ rank lower bounds in CP ([6]) and $O(\log^2 n)$ upper bounds for the depth in SP ([2]), this proves an incomparability between the two measures.
2. An almost linear lower bound for the size of SP proofs of the PHP_n^m and for Tseitin contradictions $\mathsf{Ts}(G, \omega)$ over the complete graph. These lower bounds immediately give an optimal $\Omega(\log n)$ lower bound for the depth of SP proofs of the corresponding principles.
3. A superlinear lower bound for the size of SP proofs of $\mathsf{Ts}(G, \omega)$, when G is a $n \times n$ grid graph H_n. In turn this implies an $\Omega(\log n)$ lower bound for the depth of SP proofs of $\mathsf{Ts}(H_n, \omega)$. Proofs of depth $O(\log^2 n)$ for $\mathsf{Ts}(H_n, \omega)$ are given in [2].
4. Finally we prove a linear lower bound for the size and a $\Omega(\log n)$ lower bound for the depth for the the Linear Ordering Principle LOP_n.

Our results are derived from the following initial geometrical observation: let $\mathbb{S}$ be a space of *admissible points* in $\{0, 1, 1/2\}^n$ satisfying a given unsatisfiable system of integer linear inequalities $\mathcal{F}(x_1, \ldots, x_n)$. In a SP proof for $\mathcal{F}$, at each branch $Q = (\mathbf{a}, b)$ the set of points in the $\mathsf{slab}(Q) = \{\mathbf{s} \in \mathbb{S} : b - 1 < \mathbf{ax} < b\}$ does not survive in $\mathbb{S}$. At the end of the proof on the leaves, where we have infeasible LP's, no point in $\mathbb{S}$ can survive the proof. So it is sufficient to find conditions such that, under the assumption that a proof of $\mathcal{F}$ is "small", even one point of $\mathbb{S}$ survives the proof. In pursuing this approach we use two methods.

The *antichain method.* Here we use a well-known bound based on Sperner's Theorem [7, 29] to upper bound the number of points in the slabs where the set of non-zero coefficients is sufficiently large. Trading between the number of such slabs and the number of points ruled out from the space $\mathbb{S}$ of admissible points, we obtain the lower bound.

We initially present the method and the $\Omega(\log n)$ lower bound on a set of unsatisfiable integer linear inequalities – the *Simple Pigeonhole Principle* (SPHP) – capturing the core of the counting argument used to prove the PHP efficiently in CP. Since SPHP_n has rank 1 CP proofs, it entails a strong separation between CP rank and SP depth. We then apply the method to PHP^m_n and to $\mathsf{Ts}(K_n, \omega)$.

The *covering method.* The antichain method appears too weak to prove size and depth lower bounds on $\mathsf{Ts}(G, w)$, when G is for example a grid or a pyramid. To solve this case, we consider another approach that we call the *covering method*: we reduce the problem of proving that one point in $\mathbb{S}$ survives from all the $\mathsf{slab}(Q)$ in a small proof of $\mathcal{F}$, to the problem that a set of polynomials which *essentially covers* the boolean cube $\{0, 1\}^n$ requires at least $\sqrt{n}$ polynomials, which is a well-known problem faced by Alon and Füredi in [1] and by Linial and Radhakrishnan in [21]. For this reduction to work we have to find a high dimensional projection of $\mathbb{S}$ covering the boolean cube and defined on variables effectively appearing in the proof. We prove that cycles of distance at least 2 in G work properly to this aim on $\mathsf{Ts}(G, \omega)$. Since the grid H_n has many such cycles, we can obtain the lower bound on $\mathsf{Ts}(H_n, \omega)$. The use of Linial and Radhakrishnan's result is not new in proof complexity. Part and Tzameret in [23], independently of us, were using this result in a completely different way from us in the proof system $\mathsf{Res}(\oplus)$ handling clauses over parity equations, and not relying on integer linear inequalities and geometrical reasoning.

We remark that while we were writing this version of the paper, Yehuda and Yehudayoff in [30] slightly improved the results of [21] with the consequence, noticed in their paper too, that our size lower bounds for $\mathsf{Ts}(G, \omega)$ over a grid graph is in fact superlinear.

The paper is organized as follows: We give the preliminary definitions in the next section and then we move to other sections: one on the lower bounds by the antichain method and the other on lower bounds by the covering method. The antichain method is presented on the formulas SPHP_n, PHP^m_n, Tseitin formulas for the complete graph and the LOP_n. The covering method is used for the Tseitin formulas for the grid graph. The lower bound for the Linear Ordering Principle, LOP_n, is deferred to the appendix.

2 Preliminaries

We use $[n]$ for the set $\{1, 2, \ldots, n\}$, $\mathbb{Z}/2$ for $\mathbb{Z} \cup (\mathbb{Z} + \frac{1}{2})$ and $\mathbb{Z}^+$ for $\{1, 2, \ldots\}$.

2.1 Proof systems

Here we recall the definition of the Stabbing Planes proof system from [2].

▶ **Definition 1.** *A* linear integer inequality *in the variables $x_1, \dots, x_n$ is an expression of the form $\sum_{i=1}^{n} a_i x_i \geq b$, where each a_i and b are integral. A set of such inequalities is said to be* unsatisfiable *if there are no* $0/1$ *assignments to the x variables satisfying each inequality simultaneously.*

Note that we reserve the term infeasible, in contrast to unsatisfiable, for (real or rational) linear programs.

▶ **Definition 2.** *Fix some variables $x_1, \dots, x_n$. A* Stabbing Planes (SP) *proof of a set of integer linear inequalities $\mathcal{F}$ is a binary tree $\mathcal{T}$, with each node labeled with a* query $(\mathbf{a}, b)$ *with $\mathbf{a} \in \mathbb{Z}^n, b \in \mathbb{Z}$. Out of each node we have an edge labeled with $\mathbf{a}x \geq b$ and the other labeled with its integer negation $\mathbf{a}x \leq b - 1$. Each leaf ℓ is labeled with a* LP *system P_ℓ made by a nonnegative linear combination of inequalities from $\mathcal{F}$ and the inequalities labelling the edges on the path from the root of $\mathcal{T}$ to the leaf ℓ.*

If $\mathcal{F}$ is an unsatisfiable *set of integer linear inequalities, $\mathcal{T}$ is a* Stabbing Planes (SP) refutation *of $\mathcal{F}$ if all the* LP*'s P_ℓ on the leaves of $\mathcal{T}$ are infeasible.*

▶ **Definition 3.** *The* slab *corresponding to a query $Q = (\mathbf{a}, b)$ is the set* $\mathsf{slab}(Q) = \{\mathbf{x} \in \mathbb{R}^n : b - 1 < \mathbf{ax} < b\}$ *satisfying neither of the associated inequalities.*

Since each leaf in a SP refutation is labelled by an infeasible LP, throughout this paper we will actually use the following geometric observation on SP proofs $\mathcal{T}$: the set of points in $\mathbb{R}^n$ must all be ruled out by a query somewhere in $\mathcal{T}$. In particular this will be true for those points in $\mathbb{R}^n$ which satisfy a set of integer linear inequalities $\mathcal{F}$ and which we call *feasible points* for $\mathcal{F}$.

▶ **Fact 4.** *The slabs associated with a* SP *refutation must cover the feasible points of $\mathcal{F}$. That is,*

$$\{\mathbf{y} \in \mathbb{R}^n : \mathbf{cy} \geq r \textit{ for all } (\mathbf{c}, r) \in \mathcal{F}\} \subseteq \bigcup_{(\mathbf{a},b) \in \mathcal{T}} \{\mathbf{x} \in \mathbb{R}^n : b - 1 < \mathbf{ax} < b\}$$

The *length* of a SP refutation is the number of queries in the proof tree. The *depth* of a SP refutation $\mathcal{T}$ is the longest root-to-leaf path in $\mathcal{T}$. The size (respectively depth) of refuting $\mathcal{F}$ in SP is the *minimum* size (respectively depth) over all SP refutations of $\mathcal{F}$. We call *bit-size* of a SP refutation $\mathcal{T}$ the total number of bits needed to represent every inequality in the refutation.

▶ **Definition 5** ([8])**.** *The* Cutting Planes (CP) *proof system is equipped with boolean axioms and two inference rules:*

Boolean Axioms	*Linear Combination*	*Rounding*
$\dfrac{}{x \geq 0}$ $\quad\dfrac{}{-x \geq -1}$	$\dfrac{\mathbf{ax} \geq c \qquad \mathbf{bx} \geq d}{\alpha\mathbf{ax} + \beta\mathbf{bx} \geq \alpha c + \beta d}$	$\dfrac{\alpha\mathbf{ax} \geq b}{\mathbf{ax} \geq \lceil b/\alpha \rceil}$

where $\alpha, \beta, b \in \mathbb{Z}^+$ and $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$. A CP refutation of some unsatisfiable set of integer linear inequalities is a derivation of $0 \geq 1$ *by the aforementioned inference rules from the inequalities in $\mathcal{F}$.*

A CP refutation is *treelike* if the directed acyclic graph underlying the proof is a tree. The *length* of a CP refutation is the number of inequalities in the sequence. The *depth* is the length of the longest path from the root to a leaf (sink) in the graph. The *rank* of a CP proof is the maximal number of rounding rules used in a path of the proof graph. The *size* of a CP refutation is the bit-size to represent all the inequalities in the proof.

2.2 Restrictions

Let $V = \{x_1, \dots, x_n\}$ be a set of n variables and let $\mathbf{a}\mathbf{x} \leq b$ be a linear integer inequality. We say that a variable x_i *appears in*, or is *mentioned by* a query $Q = (\mathbf{a}, b)$ if $a_i \neq 0$ and *does not appear* otherwise.

A *restriction* ρ is a function $\rho : D \to \{0, 1\}$, $D \subseteq V$. A restriction acts on a half-plane $\mathbf{a}\mathbf{x} \leq b$ setting the x_i's according to ρ. Notice that the variables $x_i \in D$ do not appear in the restricted half-plane.

By $\mathcal{T}\restriction_\rho$ we mean to apply the restriction ρ to all the queries in a SP proof $\mathcal{T}$. The tree $\mathcal{T}\restriction_\rho$ defines a new SP proof: if some $Q\restriction_\rho$ reduces to $0 \leq -b$, for some $b \geq 1$, then that node becomes a leaf in $\mathcal{T}\restriction_\rho$. Otherwise in $\mathcal{T}\restriction_\rho$ we simply branch on $Q\restriction_\rho$. Of course the solution space defined by the linear inequalities labelling a path in $\mathcal{T}\restriction_\rho$ is a subset of the solution space defined by the corresponding path in $\mathcal{T}$. Hence the leaves of $\mathcal{T}\restriction_\rho$ define an infeasible LP.

We work with linear integer inequalities which are a translation of families of CNFs $\mathcal{F}$. Hence when we write $\mathcal{F}\restriction_\rho$ we mean the applications of the restriction ρ to the set of linear integer inequalities defining $\mathcal{F}$.

3 The antichain method

This method is based on Sperner's theorem. Using it we can prove depth lower bounds in SP for PHP^m_n and for Tseitin contradictions $\mathsf{Ts}(K_n, \omega)$ over the complete graph. To motivate and explain the main definitions, we use as an example a simplification of the PHP^m_n, the *Simplified Pigeonhole principle* SPHP_n, which has some interest since (as we will show) it exponentially separates CP rank from SP depth.

3.1 Simplified Pigeonhole Principle

As mentioned in the Introduction, the SPHP_n intends to capture the core of the counting argument used to efficiently refute the PHP in CP.

▶ **Definition 6.** *The* SPHP_n *is the following unsatisfiable family of inequalities:*

$$\begin{array}{ll} \sum_{i=1}^n x_i \geq 2 & \\ x_i + x_j \leq 1 & \textit{for all } i \neq j \in [n] \\ 0 \leq x_i \leq 1 & \textit{for all } i \in [n]. \end{array}$$

▶ **Lemma 7.** SPHP_n *has a rank* 1 CP *refutation, for* $n \geq 3$.

Proof. Let $S := \sum_{i=1}^n x_i$ (so we have $S \geq 2$). We fix some $i \in [n]$ and sum $x_i + x_j \leq 1$ over all $j \in [n] \setminus \{i\}$ to find $S + (n-2)x_i \leq n-1$. We add this to $-S \leq -2$ to get

$$x_i \leq \frac{n-3}{n-2}$$

which becomes $x_i \leq 0$ after a single cut. We do this for every i and find $S \leq 0$ - a contradiction when combined with the axiom $S \geq 2$. ◀

It is easy to see that SPHP_n has depth $O(\log n)$, length $O(n)$ proofs in SP, either by a direct proof or appealing to the polynomial size proofs in CP of the PHP^m_n ([8]) and then using the Theorem 4.4 in [2] informally stating that "CP proofs can be balanced in SP".

▶ **Theorem 8.** *The* SPHP_n *has a* SP *refutation of size* $O(n)$ *and depth* $O(\log(n))$.

Proof. Note that no admissible point for the SPHP_n has any x_i set to 1. The SP refutation just performs a binary search looking for an x_i set to 1 – if it cannot find such an x_i, we contradict the axiom $\sum_{i=1}^{n} x_i \geq 2$,

In more detail, the root asks if $\sum_{i=1}^{n} x_i$ is at least 1 or at most 0. The at most 0 branch directly contradicts the axiom $\sum_{i=1}^{n} x_i \geq 2$ (and so terminates). The at least 1 branch asks if $\sum_{i=1}^{\lfloor n/2 \rfloor} x_i$ is again at least 1 or at most 0. If this is at most 0, we must have that $\sum_{i=\lfloor n/2 \rfloor+1}^{n} x_i \geq 1$ - in either case we have halved the range of the index of summation. ◀

We will prove that this depth bound is tight.

3.2 Sperner's Theorem

Let $\mathbf{a} \in \mathbb{R}^n$. The *width* $w(\mathbf{a})$ of $\mathbf{a}$ is the number of non-zero coordinates in $\mathbf{a}$. The width of a query $(\mathbf{a}, b)$ is $w(\mathbf{a})$, and the width of a SP refutation is the minimum width of its queries.

Let $n \in \mathbb{N}$. Fix $W \subseteq [0,1] \cap \mathbb{Q}^+$ of finite size $k \geq 2$ and insist that $0 \in W$. The W's we work with in this paper are $\{0, 1/2\}$ and $\{0, 1/2, 1\}$.

▶ **Definition 9.** *A (n, W)-word is an element in W^n.*

For two vectors $x, y \in \mathbb{R}^d$, say that $x \leq y$ *in the pointwise ordering* if $x_i \leq y_i$ for all $1 \leq i \leq d$. We consider the following extension of Sperner's theorem.

▶ **Theorem 10** ([22, 7]). *Fix any $t \geq 2, t \in \mathbb{N}$. For all $f \in \mathbb{N}$, with the pointwise ordering of $[t]^f$, any antichain has size at most $t^f \sqrt{\frac{6}{\pi(t^2-1)f}}(1 + o(1))$.*

We will use the simplified bound that any antichain $\mathcal{A}$ has size $|\mathcal{A}| \leq \frac{t^f}{\sqrt{f}}$.

▶ **Lemma 11.** *Let $\mathbf{a} \in \mathbb{Z}^n$ and $|W| = k \geq 2$. The number of (n, W)-words $\mathbf{s}$ such that $\mathbf{as} = b$, where $b \in \mathbb{Q}$, is at most $\frac{k^n}{\sqrt{w(\mathbf{a})}}$.*

Proof. Define $I_{\mathbf{a}} = \{i \in [n] : a_i \neq 0\}$. Let $\preceq$ be the partial order over $W^{I_{\mathbf{a}}}$ where $\mathbf{x} \preceq \mathbf{y}$ if $x_i \leq y_i$ for all i with $a_i > 0$ and $x_i \geq y_i$ for the remaining i with $a_i < 0$. Clearly the set of solutions (restricted to indices in $I_{\mathbf{a}}$) to $\mathbf{as} = b$ forms an antichain under $\preceq$. Noting that $\preceq$ is isomorphic to the typical pointwise ordering on $W^{I_{\mathbf{a}}}$, we appeal to Theorem 10 to upper bound the number of solutions in $W^{I_{\mathbf{a}}}$ by $\frac{k^{w(\mathbf{a})}}{\sqrt{w(\mathbf{a})}}$, each of which corresponds to at most $k^{n-w(\mathbf{a})}$ vectors in W^n. ◀

3.3 Large admissibility

A (n, W)-word s is *admissible* for an unsatisfiable set of integer linear inequalities $\mathcal{F}$ over n variables if s satisfies all constraints of $\mathcal{F}$. A set of (n, W)-words is admissible for $\mathcal{F}$ if all its elements are admissible. $\mathcal{A}(\mathcal{F}, W)$ is the set of all admissible (n, W)-words for $\mathcal{F}$.

The interesting sets W for an unsatisfiable set of integer linear inequalities $\mathcal{F}$ are those such that almost all (n, W)-words are admissible for $\mathcal{F}$. We will apply our method on sets of integer linear inequalities which are a translation of unsatisfiable CNF's generated over a given domain. Typically these formulas on a size n domain have a number of variables which is not exactly n but a function of n, $\nu(n) \geq n$. (For example, the PHP_n^{n+1} has $\nu(n) = n^2 + n$ variables.) Hence for the rest of this section we consider $\mathscr{F} := \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ as a family of sets of unsatisfiable integer linear inequalities, where $\mathcal{F}_n$ has $\nu(n) \geq n$ variables. We call $\mathscr{F}$ an *unsatisfiable family.*

Consider then the following definition (recalling that we denote $k = |W|$):

▶ **Definition 12.** *$\mathscr{F}$ is* almost full *if $|\mathcal{A}(\mathcal{F}_n, W)| \geq k^{\nu(n)} - o(k^{\nu(n)})$.*

Notice that, because of the o notation, Definition 12 might be not necessarily true for all $n \in \mathbb{N}$, but only starting from some $n_{\mathscr{F}}$. Note also that being almost full is defined relative to some W.

▶ **Definition 13.** *Given some almost full family $\mathscr{F}$ (over $\nu(n)$ variables) we let $n_{\mathscr{F}}$ be the natural number with*

$$\frac{k^{\nu(n)}}{|\mathcal{A}(\mathcal{F}_n, W)|} \leq 2 \quad \textit{for all} \quad n \geq n_{\mathscr{F}}.$$

As an example we prove SPHP is almost full (notice that in the case of SPHP_n, $\nu(n) = n$).

▶ **Lemma 14.** *SPHP_n is almost full when $W = \{0, 1/2\}$.*

Proof. Let U be the set of all (n, W)-words with at least four coordinates set to $1/2$. U is admissible for SPHP_n since inequalities $x_i + x_j \leq 1$ are always satisfied for any value in W and inequalities $x_1 + \ldots + x_n \geq 2$ are satisfied by all points in U which contain at least four $1/2$s. By a simple counting argument, in U there are at least $2^n - 4n^3 = 2^n - o(2^n)$ admissible (n, W)-words. Hence the claim. ◀

▶ **Lemma 15.** *Let $\mathscr{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ be an almost full unsatisfiable family, where $\mathcal{F}_n$ has $\nu(n)$ variables. Further let $\mathcal{T}$ be a SP refutation of $\mathcal{F}$ of width w. If $n \geq n_{\mathscr{F}}$ then $|\mathcal{T}| = \Omega(\sqrt{w})$.*

Proof. We estimate at what rate the slab of the queries in $\mathcal{T}$ rule out admissible points in U. Let ℓ be the least common multiple of the denominators in W. Every (n, W)-word x falling in the slab of some query $(\mathbf{a}, b)$ satisfies one of ℓ equations $\mathbf{a}x = b + i/\ell, 1 \leq i < \ell$ (as $\mathbf{a}$ is integral). Note that as $|W|$ is a constant independent of n, so is ℓ.

Since all the queries in $\mathcal{T}$ have width at least w, according to Lemma 11, each query in $\mathcal{T}$ rules out at most $\ell \cdot \frac{k^{\nu(n)}}{\sqrt{w}}$ admissible points. By Fact 4 no point survives at the leaves, in particular the admissible points. Then it must be that

$$|\mathcal{T}|\ell \cdot \frac{k^{\nu(n)}}{\sqrt{w}} \geq |\mathcal{A}(\mathcal{F}_n, W)| \quad \text{which means} \quad |\mathcal{T}|\ell \cdot \frac{k^{\nu(n)}}{|\mathcal{A}(\mathcal{F}_n, W)|} \geq \sqrt{w}$$

We finish by noting that, by the assumption $n \geq n_{\mathscr{F}}$, and then by Definition 13, we have $2 \geq \frac{k^{\nu(n)}}{|\mathcal{A}(\mathcal{F}_n, W)|}$, so $|\mathcal{T}| \geq \sqrt{w}/(2\ell) \in \Omega(\sqrt{w})$. ◀

3.4 Main theorem

We focus on restrictions ρ that after applied to an unsatisfiable family $\mathscr{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$, reduce the set $\mathcal{F}$ to another set in the same family.

▶ **Definition 16.** *Let $\mathscr{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ be an unsatisfiable family and c a positive constant. $\mathscr{F}$ is* c-self-reducible *if for any set V of variables, with $|V| = v < n/c$, there is a restriction ρ with domain $V' \supseteq V$, such that $\mathcal{F}_n\restriction_\rho = \mathcal{F}_{n-cv}$ (up to renaming of variables).*

Let us motivate the definition with an example.

▶ **Lemma 17.** *SPHP_n is 1-self-reducible.*

Proof. Whatever set of variables x_i, $i \in I \subset [n]$ we consider, it is sufficient to set x_i to 0 to fulfill Definition 16. ◀

▶ **Theorem 18.** *Let $\mathscr{F} := \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be a unsatisfiable set of integer linear inequalities which is almost full and c-self-reducible, for some constant c. If $\mathcal{F}_n$ defines a feasible* LP *whenever $n > n_{\mathscr{F}}$, then for n large enough, the shortest* SP *proof of $\mathcal{F}_n$ is of length $\Omega(\sqrt[4]{n})$.*

Proof. Take any SP proof $\mathcal{T}$ refuting $\mathcal{F}_n$ and fix $t = \sqrt[4]{n}$.

The proof proceeds by stages $i \geq 0$ where $\mathcal{T}_0 = \mathcal{T}$. The stages will go on while the invariant property (which at stage 0 is true since $n > n_{\mathscr{F}}$ and c a positive constant)

$$n - ict^3 > \max\{n_{\mathscr{F}}, n(1 - 1/c)\}$$

holds.

At the stage i we let $\Sigma_i = \{(\mathbf{a}, b) \in \mathcal{T}_i : w(\mathbf{a}) \leq t^2\}$ and $s_i = |\Sigma_i|$. If $s_i \geq t$ the claim is trivially proven. If $s_i = 0$, then all queries in $\mathcal{T}_i$ have width at least t^2 and by Lemma 15 (which can be applied since $n - ict^3 > n_{\mathscr{F}}$) the claim is proven (for n large enough).

So assume that $0 < s_i < t$. Each of the queries in Σ_i involves at most t^2 nonzero coefficients, hence in total they mention at most $s_it^2 \leq t^3$ variables. Extend this set of variables to some V' in accordance with Definition 16 (which can be done since, by the invariant, $ict^3 < n/c$). Set all these variables according to self-reducibility of $\mathcal{F}$ in a restriction ρ_i and define $\mathcal{T}_{i+1} = \mathcal{T}_i\restriction_{\rho_i}$. Note that by Definition 16 and by that of restriction, $\mathcal{T}_{i+1}$ is a SP refutation of $\mathcal{F}_{n-ict^3}$ and we can go on with the next stage. (Also note that we do not hit an empty refutation this way, due to the assumption that $\mathcal{F}_n$ defines a feasible LP.)

Assume that the invariant does not hold. If this is because $n - ict^3 < n_{\mathscr{F}}$ then, as each iteration destroys at least one node,

$$|\mathcal{T}| \geq i > \frac{n - n_{\mathscr{F}}}{ct^3} \in \Omega(n^{1/4}).$$

If this is because $n - ict^3 < n - n/c$, then again for the same reason it holds that

$$|\mathcal{T}| \geq i > \frac{n}{c^2 n^{3/4}} \in \Omega(n^{1/4}). \qquad ◀$$

Using Lemmas 14 and 17 and the previous Theorem we get:

▶ **Corollary 19.** *The length of any* SP *refutation of* SPHP_n *is $\Omega(\sqrt[4]{n})$. Hence the minimal depth is $\Omega(\log n)$.*

3.5 Lower bounds for the Pigeonhole principle

▶ **Definition 20.** *The* Pigeonhole principle PHP^m_n*, $m(n) > n$, is the family of unsatisfiable integer linear inequalities defined over the variables $\{P_{i,j} : i \in [m], j \in [n]\}$ consisting of the following inequalities:*

$$\begin{array}{lll} \sum_{j=1}^n P_{i,j} \geq 1 & \forall i \in [m] & \textit{(every pigeon goes into some hole)} \\ P_{i,k} + P_{j,k} \leq 1 & \forall k \in [n], i \neq j \in [m] & \textit{(at most one pigeon enters any given hole)} \end{array}$$

We present a lower bound for PHP^m_n closely following that for SPHP_n, in which we largely ignore the diversity of different pigeons (which makes the principle rather like SPHP_n).

In this subsection we fix $W = \{0, 1/2\}$, and for the sake of brevity refer to (n, W)-words as *biwords.*

In this section we fix m to be $n + d$, for any fixed $d \in \mathbb{N}$ at least one.

▶ **Lemma 21.** *The* PHP^{n+d}_n *is almost full.*

Proof. We show that there are at least 2^{mn-1} admissible biwords (for sufficiently large n). For each pigeon i, there are admissible valuations to holes so that, so long as at least two of these are set to $1/2$, the others may be set to anything in $\{0, 1/2\}$. This gives at least $2^n - (n+1)$ possibilities. Since the pigeons are independent, we obtain at least $(2^n - (n+1))^m$ biwords. Now this is $2^{mn}\left(1 - \frac{n+1}{2^n}\right)^m$ where $\left(1 - \frac{n+1}{2^n}\right)^m \sim e^{\frac{-(n+1)m}{2^n}}$ whence, $\left(1 - \frac{n+1}{2^n}\right)^m \geq e^{\frac{-(n+2)m}{2^n}}$ for sufficiently large n. It follows there is a constant c so that:

$$2^{mn}\left(1 - \frac{n+1}{2^n}\right)^m \geq 2^{mn - \frac{c(n+2)m}{2^n}} \geq 2^{mn-1}$$

for sufficiently large n. ◀

▶ **Lemma 22.** *The* PHP^{n+d}_n *is* 1*-self-reducible.*

Proof. We are given some set I of variables from PHP^{n+d}_n. These variables will mention some set of holes $H := \{j : P_{i,j} \in I \text{ for some i}\}$ and similarly a set of pigeons P. Each of P, H have size at most $|I|$ and we extend them both arbitrarily to have size exactly $|I|$. Our restriction matches P and H in any way and then sets any other variable mentioning a pigeon in P or a hole in H to 0. ◀

▶ **Theorem 23.** *The length of any* SP *refutation of* PHP^{n+d}_n *is* $\Omega(\sqrt[4]{n})$*.*

Proof. Note that the all 1/2 point is feasible for PHP^{n+d}_n. Then with Lemma 21 and Lemma 22 in hand we meet all the prerequisites for Theorem 18. ◀

By simply noting that a SP refutation is a binary tree, we get the following corollary.

▶ **Corollary 24.** *The* SP *depth of the* PHP^{n+d}_n *is* $\Omega(\log n)$*.*

3.6 Lower bounds for Tseitin contradictions over the complete graph

▶ **Definition 25.** *For a graph* $G = (V, E)$ *along with a charging function* $\omega : V \to \{0, 1\}$ *satisfying* $\sum_{v \in V} \omega(v) = 1 \mod 2$*. The* Tseitin contradiction $\mathsf{Ts}(G, \omega)$ *is the set of linear inequalities which translate the CNF encoding of*

$$\sum_{\substack{e \in E \\ e \ni v}} x_e = \omega(v) \mod 2.$$

for every $v \in V$*, where the variables* x_e *range over the edges* $e \in E$*.*

In this subsection we consider $\mathsf{Ts}(K_n, \omega)$ and ω will always be an odd charging for K_n. We let $N := \binom{n}{2}$ and we fix $W = \{0, 1/2, 1\}$, $k = 3$ and for the sake of brevity refer to (n, W)-words as *triwords*. We will abuse slightly the notation of Section 3.3 and consider the family $\{\mathsf{Ts}(K_n, \omega)\}_{n \in \mathbb{N},\, \omega \text{ odd}}$ as a single parameter family in n. The reason we can do this is because the following proofs of almost fullness and self reducibility do not depend on ω at all (so long as it is odd, which we will always ensure).

▶ **Lemma 26.** $\mathsf{Ts}(K_n, \omega)$ *is almost full.*

Proof. We show that $\mathsf{Ts}(K_n, \omega)$ has at least $c3^N$ admissible triwords, for any constant $0 < c < 1$ and n large enough. We define the assignment ρ setting all edges (i.e. x_e) to a value in $W = \{0, 1, 1/2\}$ independently and uniformly at random, and inspecting the probability that some fixed constraint for a node v is violated by ρ.

Clearly if at least 2 edges incident to v are set to $1/2$ its constraint is satisfied. If none of its incident edges are set to $1/2$ then it is satisfied with probability $1/2$. Let $A(v)$ be the event "*no edge incident to v is set to $1/2$ by ρ*" and let $B(v)$ be the event that "*exactly one edge incident to v is set to $1/2$ by ρ*". Then:

$$\Pr[v \text{ is violated}] \leq \frac{1}{2}\Pr[A(v)] + \Pr[B(v)] = \frac{1}{2}\frac{2^{n-1}}{3^{n-1}} + \frac{(n-1)2^{n-2}}{3^{n-1}} = n\frac{2^{n-2}}{3^{n-1}}.$$

Therefore, by a union bound, the probability that there exists a node with violated parity is bounded above by $n^2\frac{2^{n-2}}{3^{n-1}}$, which approaches 0 as n goes to infinity. ◀

▶ **Lemma 27.** *$\mathsf{Ts}(K_n, \omega)$ is 2-self-reducible.*

Proof. We are given some set of variables I. Each variable mentions 2 nodes, so extend these mentioned nodes arbitrarily to a set S of size exactly $2|I|$, which we then hit with the following restriction: if S is evenly charged, pick any matching on the set $\{s \in S : w(s) = 1\}$, set those edges to 1, and set any other edges involving some vertex in S to 0. Otherwise (if S is oddly charged) pick any $l \in \{s \in S : w(s) = 1\}$ and $r \in [n] \setminus S$ and set x_{lr} to 1. $\{s \in S : w(s) = 1\} \setminus l$ is now even so we can pick a matching as before. And as before we set all other edges involving some vertex in S to 0. In the first case the graph induced by $[n] \setminus S$ must be oddly charged (as the original graph was). In the second case this induced graph was originally evenly charged, but we changed this when we set x_{lr} to 1. ◀

▶ **Lemma 28.** *For any oddly charged ω and n large enough, all* SP *refutations of $\mathsf{Ts}(K_n, \omega)$ have length $\Omega(\sqrt[4]{n})$.*

Proof. We have that the all 1/2 point is feasible for $\mathsf{Ts}(K_n, \omega)$. Then we can simply apply Theorem 18. ◀

▶ **Corollary 29.** *The depth of any SP refutation of $\mathsf{Ts}(K_n, \omega)$ is $\Omega(\log n)$.*

4 The covering method

▶ **Definition 30.** *A set L of linear polynomials with real coefficients is said to be a* cover *of the cube $\{0, 1\}^n$ if for each $v \in \{0, 1\}^n$, there is a $p \in L$ such that $p(v) = 0$.*

In [21] Linial and Radhakrishnan considered the problem of the minimal number of hyperplanes needed to cover the cube $\{0, 1\}^n$. Clearly every such cube can be covered by the zero polynomial, so to make the problem more meaningful they defined the notion of an *essential covering* of $\{0, 1\}^n$.

▶ **Definition 31** ([21])**.** *A set L of linear polynomials with real coefficients is said to be an* essential cover *of the cube $\{0, 1\}^n$ if*

(E1) *L is a cover of $\{0, 1\}^n$,*
(E2) *no proper subset of L satisfies (E1), that is, for every $p \in L$, there is a $v \in \{0, 1\}^n$ such that p alone takes the value 0 on v, and*
(E3) *every variable appears (in some monomial with non-zero coefficient) in some polynomial of L.*

They then proved that any essential cover E of the hypercube $\{0, 1\}^n$ must satisfy $|E| \geq \sqrt{n}$. We will use the slightly strengthened lower bound given in [31]:

▶ **Theorem 32.** *Any essential cover L of the cube with n coordinates satisfies $|L| \in \Omega(n^{0.52})$.*

We will need an auxillary definition and lemma.

▶ **Definition 33.** *Let L be a cover of $\{0,1\}^I$ for some index set I. Some subset L' of L is an* essentialisation *of L if L' also covers $\{0,1\}^I$ but no proper subset of it does.*

▶ **Lemma 34.** *Let L be a cover of the cube $\{0,1\}^n$ and L' be any essentialisation of L. Let M' be the set of variables appearing with nonzero coefficient in L'. Then L' is an essential cover of $\{0,1\}^{M'}$.*

Proof.

(E1) Given any point $x \in \{0,1\}^{M'}$, we can extend it arbitrarily to a point $x' \in \{0,1\}^M$. Then there is some $p \in L'$ with $p(x') = 0$ – but $p(x') = p(x)$, as p doesn't mention any variable outside of M'.

(E2) Similarly to the previous point, this will follow from the fact that if some set $\mathcal{T}$ covers a hypercube $\{0,1\}^I$, it also covers $\{0,1\}^{I'}$ for any $I' \supseteq I$.
Suppose some proper subset $L'' \subset L'$ covers $\{0,1\}^{M'}$, then it covers $\{0,1\}^n$ – but we picked L' to be a minimal set with this property.

(E3) We defined M' to be the set of variables appearing with nonzero coefficient in L'. ◀

4.1 The covering method and Tseitin

Let H_n denote the $n \times n$ grid graph. Fix some ω with odd charge and a SP refutation $\mathcal{T}$ of $\mathsf{Ts}(H_n, \omega)$. Fact 4 tells us that for every point x admissible for $\mathsf{Ts}(H_n, \omega)$, there exists a query $(\mathbf{a}, b) \in \mathcal{T}$ such that $b < \mathbf{a}x < b + 1$. In this section we will only consider admissible points with entries in $\{0, 1/2, 1\}$, turning the slab of a query $(\mathbf{a}, b)$ into the solution set of the single linear equation $\mathbf{a} \cdot x = b + 1/2$. So we consider $\mathcal{T}$ as a set of such equations.

We say that an edge of H_n is *mentioned* in $\mathcal{T}$ if the variable x_e appears with non-zero coefficient in some query in $\mathcal{T}$. We can see H_n as a set of $(n-1)^2$ squares (4-cycles), and we can index them as if they were a Cartesian grid, starting from 1. Let S be the set of $\lfloor (n/3)^2 \rfloor$ squares in H_n gotten by picking squares with indices that become 2 (mod 3). This ensures that every two squares in S in the same row or column have at least two other squares between them, and that no selected square is on the perimeter.

We will assume WLOG that n is a multiple of 3, so $|S| = (n/3)^2$. Let $K = \bigcup_{t \in S} t$ be the set of edges mentioned by S, and for some $s \in S$, let $K_s := \bigcup_{t \in S, t \neq s} t$ be the set of edges mentioned in S by squares other than s.

▶ **Lemma 35.** *For every $s \in S$ we can find an admissible point $b_s \in \{0, 1/2, 1\}^{E(H_n)}$ such that*

1. *$b_s(x_e) = 0$ for all $e \in K_s$, and*
2. *b_s is fractional only on the edges in s.*

Proof. We use the following fact due to A. Urquhart in [28]

▶ **Fact 36.** *For each vertex v in H_n there is a totally binary assignment, called v-critical in [28], satisfying all parity axioms in $\mathsf{Ts}(H_n, \omega)$ except the parity axiom of node v.*

Pick any corner c of s. Let b_s be the result of taking any c-critical assignment of the variables of $\mathsf{Ts}(H_n, \omega)$ and setting the edges in s to 1/2. b_s is admissible, as c is now adjacent to two variables set to 1/2 (so its originally falsified parity axiom becomes satisfied) and every other vertex is either unaffected or also adjacent to two 1/2s. While b_s sets some edge $e \in K_s$ to 1, flip all of the edges in the unique other square containing e. This other square always exists (as no square touches the perimeter) and also contains no other edge in K_s (as there are at least two squares between any two squares in S). Flipping the edges in a cycle preserves admissibility, as every vertex is adjacent to 0 or 2 flipped edges. ◀

▶ **Definition 37.** *Let $V_S := \{v_s : s \in S\}$ be a set of new variables. For $s \in S$ define the substitution h_s, taking the variables of $\mathsf{Ts}(H_n, \omega)$ to $V_S \cup \{0, 1/2, 1\}$, as*

$$h_s(x_e) := \begin{cases} b_s(e) & \textit{if } e \textit{ is not mentioned in } S\textit{, or if } e \textit{ is mentioned by } s, \\ v_t & \textit{if } e \textit{ is mentioned by some square } t \neq s \in S. \end{cases}$$

(where b_s is from Lemma 35).

▶ **Definition 38.** *Say that a linear polynomial $p = c + \sum_{e \in E(H_n)} \mu_e x_e$ with coefficients $\mu_e \in \mathbb{Z}$ and some constant part $c \in \mathbb{R}$ has* odd coefficient *in $X \subseteq E(H_n)$ if $\sum_{e \in X} \mu_e$ is an odd integer. Given some polynomial p in the variables x_e of Tseitin, and some square $s \in S$, let p_s be the polynomial in variables V_S gotten by applying the substitution $x_e \to h_s(x_e)$. Also, for any set of polynomials $\mathcal{T}$ in the variables x_e let $\mathcal{T}_s := \{p_s : p \in \mathcal{T}, p$ has odd coefficient in $s\}$.*

Given some assignment $\alpha \in \{0,1\}^{V_S \setminus \{v_s\}}$, and some h_s as in Definition 37, we let $\alpha(h_s)$ be the assignment to the variables of $\mathsf{Ts}(H_n, \omega)$ gotten by replacing the v_t in the definition of h_s by $\alpha(v_t)$.

▶ **Lemma 39.** *Let $s \in S$. For all $2^{|S|-1}$ settings α of the variables in $V_S \setminus \{v_s\}$, $\alpha(h_s)$ is admissible.*

Proof. When $\alpha(v_t)$ is all 0, $h_s = b_s$ is admissible (by Lemma 35). Toggling some v_t only has the effect of flipping every edge in a cycle, which preserves admissibility. ◀

▶ **Lemma 40.** *$\mathcal{T}_s$ covers $\{0,1\}^{V_S \setminus \{v_s\}}$.*

Proof. For every setting of $\alpha \in \{0,1\}^{V_S \setminus \{v_s\}}$, $\alpha(h_s)$ as defined above is admissible and therefore covered by some $p \in \mathcal{T}$, which has constant part $1/2 + b$ for some $b \in \mathbb{Z}$. Furthermore, as $\alpha(h_s)$ sets every edge in s to $1/2$, every such p must have odd coefficient in front of s - otherwise

$$p(\alpha(h_s)) = 1/2 + b + (1/2)\left(\sum_{e \in s} \mu_e\right) + \sum_{e \notin s} \mu_e \alpha(h_s)(x_e)$$

can never be zero, as the $1/2$ is the only non integral term in the summation. ◀

▶ **Theorem 41.** *Any SP refutation $\mathcal{T}$ of $\mathsf{Ts}(H_n, \omega)$ must have $|\mathcal{T}| \in \Omega(n^{1.04})$.*

Proof. We are going to find a set of pairs $(L_1, M_1), (L_2, M_2), \ldots, (L_q, M_q)$, where the L_i are pairwise disjoint nonempty subsets of $\mathcal{T}$, the M_i are subsets of V_S, and for every i there is some $s_i \in S \setminus \bigcup_{i=1}^q M_i$ such that $|(L_i)_{s_i}| \geq |M_i|^{0.52}$. These pairs will also satisfy the property that

$$\{s_i : 1 \leq i \leq q\} \cup \bigcup_{i=1}^{q} M_i = S. \tag{1}$$

As $|S| = (n/3)^2$ this would imply that $\sum_{i=1}^q |M_i| \geq (n/3)^2 - q$. If $q \geq (n/3)^2/2$, then (as the L_i are nonempty and pairwise disjoint) we have $|\mathcal{T}| \geq (n/3)^2/2 \in \Omega(n^{1.04})$. Otherwise $\sum_{i=1}^q |M_i| \geq (n/3)^2/2$, and as (by Theorem 32) each $|L_i| \geq |M_i|^{0.52}$,

$$|\mathcal{T}| \geq \sum_{i=1}^{q} |L_i| \geq \sum_{i=1}^{q} |M_i|^{0.52} \geq \left(\sum_{i=1}^{q} |M_i|\right)^{0.52} \geq \left((n/3)^2/2\right)^{0.52} \in \Omega(n^{1.04}). \tag{2}$$

We create the pairs by stages. Let $S_1 = S$ and start by picking any $s_1 \in S_1$. By Lemma 40 $\mathcal{T}_{s_1}$ covers $\{0,1\}^{V_{S_1}\setminus\{v_{s_1}\}}$ and has as an essentialisation E, which will be an essential cover of $\{0,1\}^{V'}$ for some $V' \subseteq V_{S_1} \setminus \{v_{s_1}\}$. We create the pair $(L_1, M_1) = (\{p : p_{s_1} \in E\}, V')$ and update $S_2 = S_1 \setminus (\{s : v_s \in V'\} \cup \{s_1\})$. (Note that V' could possibly be empty - for example, if the polynomial $x_e = 1/2$ appears in $\mathcal{T}$, where $e \in s_1$. In this case however we still have $|L_1| \geq |M_1|^{0.52}$. If V' is not empty we have the same bound due to Theorem 32.) If S_2 is nonempty we repeat with any $s_2 \in S_2$, and so on.

We now show that as promised the left hand sides of these pairs partition a subset of $\mathcal{T}$, which will give us the first inequality in Equation (2). Every polynomial p with $p \in L_i$ has every v_t mentioned by p_{s_i} removed from S_j for all $j \geq i$, so the only way p could reappear in some later L_j is if $p_{s_j} \in \mathcal{T}_{s_j}$, where v_{s_j} does *not* appear in p_{s_i}. Let $\mu_e, e \in s_j$ be the coefficients of p in front of the four edges of s_j. The coefficient in front of v_{s_j} in p_{s_i} is just $\sum_{e \in s_j} \mu_e$. As v_{s_j} failed to appear this sum is 0 and p does not have the odd coefficient sum it would need to appear in $\mathcal{T}_{s_j}$. ◀

5 Conclusions

The $\Omega(\log n)$ depth lower bound for $\mathsf{Ts}(H_n, \omega)$ is not optimal since [2] proved an $O(\log^2 n)$ upper bound for $\mathsf{Ts}(G, \omega)$, for any bounded-degree G. Even to apply the covering method to prove a depth $\Omega(\log^2 n)$ lower bound on $\mathsf{Ts}(K_n, \omega)$ (notice that it would imply a superpolynomial length lower bound), the polynomial covering of the boolean cube should be improved to work on general cubes. To this end the algebraic method used in [21] should be improved to work with generalizations of multilinear polynomials.

One weakness of the lower bound techniques presented in this work is that we consider coverings of polytopes with slabs, rather than *recursive* coverings. That is, if we branch on $ax \geq b$ then any further query that we do on the branch $ax \leq b - 1$ will not affect the points on the branch $ax \geq b$. Thus our method is overstating the number of points ruled out by each slab. In treelike proof systems where proofs can be balanced and depth lower bounds give size lower bounds [24, 3, 4, 23] such a recursive method can be approached through the Prover-Delayer game-theoretic tool [24] or generalizations of this game [4, 5]. Proving stronger direct lower bounds on Stabbing Planes by recursive methods is a direction for further research left open in this work.

References

1 Noga Alon and Zoltán Füredi. Covering the cube by affine hyperplanes. *Eur. J. Comb.*, 14(2):79–83, 1993. doi:10.1006/eujc.1993.1011.

2 Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing planes. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.10.

3 Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Comb.*, 24(4):585–603, 2004. doi:10.1007/s00493-004-0036-5.

4 Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Inf. Process. Lett.*, 110(23):1074–1077, 2010. doi:10.1016/j.ipl.2010.09.007.

5 Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like resolution size. *Inf. Process. Lett.*, 113(18):666–671, 2013. doi:10.1016/j.ipl.2013.06.002.

6 Joshua Buresh-Oppenheim, Nicola Galesi, Shlomo Hoory, Avner Magen, and Toniann Pitassi. Rank bounds and integrality gaps for cutting planes procedures. *Theory of Computing*, 2(4):65–90, 2006.

7 Teena Carroll, Joshua Cooper, and Prasad Tetali. Counting antichains and linear extensions in generalizations of the boolean lattice, 2009.

8 W. Cook, C. R. Coullard, and G. Turán. On the complexity of cutting-plane proofs. *Discrete Appl. Math.*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4.

9 Daniel Dadush and Samarth Tiwari. On the complexity of branching proofs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 34:1–34:35. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.34.

10 Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. doi:10.1145/368273.368557.

11 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960. doi:10.1145/321033.321034.

12 Noah Fleming, Mika Göös, Russell Impagliazzo, Toniann Pitassi, Robert Robere, Li-Yang Tan, and Avi Wigderson. On the power and limitations of branch and cut. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 6:1–6:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.6.

13 Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random Θ(log n)-cnfs are hard for cutting planes. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 109–120. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.19.

14 Nicola Galesi, Pavel Pudlák, and Neil Thapen. The space complexity of cutting planes refutations. In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 433–447. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CCC.2015.433.

15 Pavel Hrubes and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 121–131. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.20.

16 Russell Impagliazzo, Toniann Pitassi, and Alasdair Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 220–228. IEEE, 1994.

17 Henry A. Kautz and Bart Selman. Ten challenges redux: Recent progress in propositional reasoning and search. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003. doi:10.1007/978-3-540-45193-8_1.

18 Arist Kojevnikov. Improved lower bounds for tree-like resolution over linear inequalities. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 70–79. Springer, 2007. doi:10.1007/978-3-540-72788-0_10.

19 Jan Krajícek. Discretely ordered modules as a first-order extension of the cutting planes proof system. *J. Symb. Log.*, 63(4):1582–1596, 1998. doi:10.2307/2586668.

20 Jan Krajícek. Interpolation by a game. *Math. Log. Q.*, 44:450–458, 1998. doi:10.1002/malq.19980440403.

21 Nathan Linial and Jaikumar Radhakrishnan. Essential covers of the cube by hyperplanes. *Journal of Combinatorial Theory, Series A*, 109(2):331–338, 2005.

22 Lutz Mattner and Bero Roos. Maximal probabilities of convolution powers of discrete uniform distributions. *Statistics & probability letters*, 78(17):2992–2996, 2008.

23 Fedor Part and Iddo Tzameret. Resolution with counting: Dag-like lower bounds and different moduli. *Comput. Complex.*, 30(1):2, 2021. doi:10.1007/s00037-020-00202-x.

24 Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-sat (preliminary version). In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*, pages 128–136. ACM/SIAM, 2000. URL: `http://dl.acm.org/citation.cfm?id=338219.338244`.

25 Mark Nicholas Charles Rhodes. On the chvátal rank of the pigeonhole principle. *Theor. Comput. Sci.*, 410(27-29):2774–2778, 2009. `doi:10.1016/j.tcs.2009.03.035`.

26 John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965. `doi:10.1145/321250.321253`.

27 Bart Selman, Henry A. Kautz, and David A. McAllester. Ten challenges in propositional reasoning and search. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 50–54. Morgan Kaufmann, 1997. URL: `http://ijcai.org/Proceedings/97-1/Papers/008.pdf`.

28 Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. `doi:10.1145/7531.8928`.

29 Jacobus Hendricus van Lint and Richard Michael Wilson. *A Course in Combinatorics*. Cambridge University Press, Cambridge, U.K.; New York, 2001.

30 Gal Yehuda and Amir Yehudayoff. A lower bound for essential covers of the cube. *CoRR*, abs/2105.13615, 2021. `arXiv:2105.13615`.

31 Gal Yehuda and Amir Yehudayoff. A lower bound for essential covers of the cube. *arXiv preprint*, 2021. `arXiv:2105.13615`.

A Lower bound for the Least Ordering Principle

▶ **Definition 42.** *Let $n \in \mathbb{N}$. The* Least Ordering Principle, LOP_n, *is the following set of unsatisfiable linear inequalities over the variables $P_{i,j}$ $(i \neq j \in [n])$:*

$$P_{i,j} + P_{j,i} = 1 \quad \textit{for all } i \neq j \in [n]$$
$$P_{i,k} - P_{i,j} - P_{j,k} \geq 1 \textit{ for all } i \neq j \neq k \in [n]$$
$$\sum_{i=1, i\neq j}^{n} P_{i,j} \geq 1 \textit{ for all } j \in [n]$$

▶ **Lemma 43.** *For any $X \subseteq [n]$ of size at most $n-3$, there is an admissible point for LOP_n integer on any edge mentioning an element in X.*

Proof. Let $\preceq$ be any total order on the elements in X. Our admissible point x will be

$$x(P_{i,j}) = \begin{cases} 1 & \text{if } i,j \in X \text{ and } i \preceq j, \text{ or if } i \notin X, j \in X \\ 0 & \text{if } i,j \in X \text{ and } j \preceq i, \text{ or if } i \in X, j \notin X \\ 1/2 & \text{otherwise (if } i,j \notin X). \end{cases}$$

The existential axioms $\sum_{i=1,i\neq j}^{n} P_{i,j}$ are always satisfied - if $j \in X$ then there is some $i \notin X$ with $P_{i,j} = 1$, and otherwise there are at least two distinct $i,k \neq j \in X$ with $P_{i,j}, P_{k,j} = 1/2$. For the transitivity axioms $P_{i,k} - P_{i,j} - P_{j,k} \geq 1$, note that if 2 or more of i,j,k are not in X there are at least 2 variables set to $1/2$, and otherwise it is set in a binary fashion to something consistent with a total order. ◀

We will assume that a SP refutation $\mathcal{T}$ of LOP_n only involves variables $P_{i,j}$ where $i < j$ - this is without loss of generality as we can safely set $P_{j,i}$ to $1 - P_{i,j}$ whenever $i > j$, and will often write $P_{\{i,j\}}$ for such a variable. We consider the underlying graph of the support of a query, i.e. an undirected graph with edges $\{i,j\}$ for every variable $P_{\{i,j\}}$ that appears with non-zero coefficient in the query.

For some function $f(n)$, we say the query is $f(n)$-*wide* if the smallest edge cover of its graph has at least $f(n)$ nodes . A query that is not $f(n)$-wide is $f(n)$-*narrow.* The next lemma works much the same as Theorem 18.

▶ **Lemma 44.** *Fix $\epsilon > 0$ and suppose we have some* SP *refutation $\mathcal{T}$ of* LOP_n*, where $|\mathcal{T}| \leq n^{\frac{1-\epsilon}{4}}$. Then, if n is large enough, we can find some* SP *refutation $\mathcal{T}'$ of* LOP_{cn}*, where c is a positive universal constant that may be taken arbitrarily close to* 1*, $\mathcal{T}'$ contains only $n^{3/4}$-wide queries, and $|T'| \leq |\mathcal{T}|$.*

Proof. We iteratively build up an initially empty restriction ρ. At every stage ρ imposes a total order on some subset $X \subseteq [n]$ and places the elements in X above the elements not in X. So ρ sets every edge not contained entirely in $[n] \setminus X$ to something binary, and $\mathsf{LOP}_n\restriction_\rho = \mathsf{LOP}_{n-|X|}$ (up to a renaming of variables).

While there exists a $n^{3/4}$-narrow query $q \in \mathcal{T}\restriction_\rho$ we simply take its smallest edge cover, which has size at most $n^{3/4}$ by definition, and add its nodes in any fashion to the total order in ρ. Now all of the variables mentioned by $q \in \mathcal{T}\restriction_\rho$ are fully evaluated and q is redundant. We repeat this at most $n^{\frac{1-\epsilon}{4}}$ times (as $|\mathcal{T}| \leq n^{\frac{1-\epsilon}{4}}$ and each iteration renders at least one query in $\mathcal{T}$ redundant). At each stage we grow the domain of the restriction by at most $n^{3/4}$, so the domain of ρ is always bounded by $n^{1-\epsilon/4}$. We also cannot exhaust the tree $\mathcal{T}$ in this way, as otherwise $\mathcal{T}$ mentioned at most $n^{1-\epsilon/4} < n-3$ elements and by Lemma 43 there is an admissible point not falling in any slab of $\mathcal{T}$, violating Fact 4.

When this process finishes we are left with a $n^{3/4}$-wide refutation $\mathcal{T}'$ of $\mathsf{LOP}_{n-n^{1-\epsilon/4}}$. As ϵ was fixed we find that as n goes to infinity $n - n^{1-\epsilon/4}$ tends to n. ◀

▶ **Lemma 45.** *Let $d \leq (n-3)/2$. Given any disjoint set of pairs $D = \{\{l_1, r_1\}, \ldots, \{l_d, r_d\}\}$ (where WLOG $l_i < r_i$ in $[n]$ as natural numbers) and any binary assignment $b \in \{0,1\}^D$, the assignment x_b with*

$$x_b(P_{\{i,j\}}) = \begin{cases} b(\{l_k, r_k\}) & \text{if } \{i,j\} = \{l_k, r_k\} \in X \text{ for some } k \\ 1/2 & \text{otherwise} \end{cases}$$

is admissible.

Proof. The existential axioms $\sum_{i=1, i\neq j}^{n} P_{i,j}$ are always satisfied, as for any j there are at least $n-2$ $i \in [n]$ different from j with $P_{i,j} = 1/2$. For the transitivity axioms $P_{i,k} - P_{i,j} - P_{j,k} \geq 1$, note that due to the disjointness of D at least two variables on the left hand side are set to $1/2$. ◀

▶ **Theorem 46.** *Fix some $\epsilon > 0$ and let $\mathcal{T}$ any* SP *refutation of* LOP_n*. Then, for n large enough, $|\mathcal{T}| \in \Omega(n^{\frac{1-\epsilon}{4}})$.*

Proof. Suppose otherwise - then, by Lemma 44, we can find some $\mathcal{T}'$ refuting LOP_{cn}, with $|\mathcal{T}'| \leq |\mathcal{T}|$, every query $n^{3/4}$-wide, and c independent of n. We greedily create a set of pairs D by processing the queries in $\mathcal{T}'$ one by one and choosing in each a matching of size $n^{1/2}$ disjoint from the elements appearing in D - this always succeeds, as at every stage $|D| \in O(n^{\frac{1-\epsilon}{4}} \cdot n^{1/2})$ and involves at most $O(2n^{\frac{3-\epsilon}{4}}) < n^{3/4} - n^{1/2}$ elements.

So by Lemma 45, after setting every edge not in D to $1/2$, we have some set of linear polynomials $\mathcal{R} = \{a(x) = \mathbf{a}x - b - 1/2 : (\overline{a}, b) \in \mathcal{T}'\}$ covering the hypercube $\{0,1\}^D$, where every polynomial $p \in \mathcal{R}$ mentions at least $n^{1/2}$ edges. By Lemma 11 each such polynomial in $\mathcal{R}$ rules out at most ${}^{2^{|D|}}/_{n^{1/4}}$ points, and so we must have $|\mathcal{T}| \geq |T'| \geq |\mathcal{R}| \geq n^{1/4}$. ◀

Linear Space Data Structures for Finite Groups with Constant Query-Time

Bireswar Das[1]
Indian Institute of Technology Gandhinagar, India

Anant Kumar
Indian Institute of Technology Gandhinagar, India

Shivdutt Sharma
Indian Institute of Information Technology, Una, India

Dhara Thakkar
Indian Institute of Technology Gandhinagar, India

Abstract

A finite group of order n can be represented by its Cayley table. In the word-RAM model the Cayley table of a group of order n can be stored using $O(n^2)$ words and can be used to answer a multiplication query in constant time. It is interesting to ask if we can design a data structure to store a group of order n that uses $o(n^2)$ space but can still answer a multiplication query in constant time.

We design a constant query-time data structure that can store any finite group using $O(n)$ words where n is the order of the group.

Farzan and Munro (ISSAC 2006) gave an information theoretic lower bound of $\Omega(n)$ on the number of words to store a group of order n. Since our data structure achieves this lower bound and answers queries in constant time, it is optimal in both space usage and query-time.

A crucial step in the process is essentially to design linear space and constant query-time data structures for nonabelian simple groups. The data structures for nonableian simple groups are designed using a lemma that we prove using the Classification Theorem for Finite Simple Groups (CFSG).

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Data compression

Keywords and phrases Compact Data Structures, Space Efficient Representations, Finite Groups, Simple Groups, Classification Theorem for Finite Simple Groups

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.25

Funding *Bireswar Das*: Funded by Ministry of Education, Govt. of India.
Dhara Thakkar: Funded by CSIR-UGC NET JRF Fellowship.

1 Introduction

The Cayley table of a group of order n is a two dimensional table whose (i, j)th entry is the product of the ith and jth element of the group. In the word-RAM model while it takes $O(n^2)$ words to store the Cayley table of a group of order n, a multiplication query can be answered in constant time by accessing the appropriate location of the table.

For many computational problems in group theory the input group is given by its Cayley table. Some of these problems include the minimum generating set problem, various problems in property testing, the group factoring problem, and the group isomorphism

[1] Corresponding author.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 25; pp. 25:1–25:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

problem [18, 1, 15, 19]. Among these, the group isomorphism problem is probably the most prominent one because of its unresolved complexity status despite years of extensive research [4, 13, 2, 5, 20, 14].

The Cayley table is very fast in terms of query processing but it takes quadratic space to store a group. It is interesting to ask if we can design a data structure for finite groups using $o(n^2)$ space[2] which can still answer multiplication query in constant time. We note that while quasigroups, and semigroups can also be stored using their Cayley tables, it is not possible to store quasigroups, and semigroups using $o(n^2)$ space. This is simply because the numbers of quasigroups, and semigroups are too large [26, 17] and the information theoretic lower bound is $\Omega(n^2 \log n)$ bits or $\Omega(n^2)$ words.

Das et al. [9] showed that for any finite group G of order n and for any $\delta \in [1/\log n, 1]$, a data structure can be constructed for G that uses $O(n^{1+\delta}/\delta)$ space and answers a multiplication query in time $O(1/\delta)$. Their result implies that there exist constant query-time data structures for finite groups of order n that use $O(n^{1.01})$ space. However, the result cannot be used to design a constant query-time data structure even if we are allowed to use $\Theta(n.polylog(n))$ space.

In this paper we design constant query-time data structures for finite groups that can be stored using $O(n)$ words where n is the order of the group. An information theoretic argument by Farzan and Munro shows that a lower bound to store a group of order n is $\Omega(n \log n)$ bits or $\Omega(n)$ words [11]. Our data structure is optimal in the sense that it achieves the lower bound. A data structure that achieves the optimum information theoretic lower bound asymptotically is known as *a compact data structure*. Therefore our data structure is a constant query-time compact data structure for finite groups. We note that compact query-time data structures were designed for some restricted classes of groups such as abelian groups and Dedekind groups [8].

In the process of designing the data structure we first prove two results, which we call *extension theorems*, on the construction of data structures for a group when we already have a data structure for a subgroup of the given group. The extra space used by the newly constructed data structure depends on the index of the subgroup in one of the results and the structure[3] of the subgroup in the other result. This indicates that finding suitable subgroups of a group might be useful.

The Jordan-Hölder theorem provides us with a supply of subgroups in the form of composition series. In our process we try to pick some suitable subgroups that are elements of the composition series of the given group. However, picking suitable groups is not always possible. This happens, as we will see in Section 4, when there is a “large” composition factor sitting in a certain position of the composition series. The composition factors are simple groups. In a sense the hard cases for constructing the data structure are for the simple groups.

Simple groups are sometimes considered as the building blocks for finite groups. The Classification Theorem for Finite Simple Groups (CFSG) is one of the most important theorems in group theory. Informally, this theorem classifies the finite simple groups into cyclic groups, alternating groups, certain groups of Lie-type and into 26 sporadic simple groups. The precise statement of the theorem could be found in Section 5. Except for the 26 sporadic simple groups the other group classes are infinite. We use CFSG to prove a key lemma that allows us to handle the case for the nonabelian simple groups.

[2] In this paper we use the word-RAM model. The space used by a data structure or an algorithm refers to the number of words used by them.

[3] The subgroup needs to be normal and quotient needs to be cyclic.

We note that for solvable groups the design of the data structure is *independent* of CFSG. The composition factors of a solvable group are cyclic of prime order. Such cases are handled using one of the extension theorems proved in Section 3.

Related work. Farzan and Munro [11] gave a succinct representations for finite abelian groups in a specific model of computation. In their model *a compression algorithm* first produces labels of each group element. The queries are processed by a *query processing unit* which is similar to the word-RAM model. However, along with the common arithmetic, logical and comparison operations the query processing unit can also perform bit-reversal in constant time. A user issuing a query, supplies the labels of two group elements that were generated by the compression algorithm to the query processing unit which then returns the label of the product of the two elements.

Das et al. [9] and Das and Sharma [8] have used Erdös-Réyni cube generating sequences, Remak-Krull-Schmidt decomposition and the structure of indecomposable groups to design their space and query-time efficient data structures. Our approach is quite different in the sense that we use the extension theorems (Section 3) and the Classification Theorem for Finite Simple groups to design the data structures.

Remark. There are several ways to represent a finite group apart from the Cayley table representation. The permutation group representation, the polycyclic presentations and the generator-relator presentations are some of the common group representations. These representations are often incomparable. For example in the generator-relator presentation we can represent infinite groups. However, many problems such as the membership testing, testing if a group is finite becomes undecidable in the generator-relator presentation (c.f. [25]). In the permutation group representation the membership testing takes superlinear time in terms of the degree of the representation and polylogarithmic in the order of the group [24, 23, 12]. We contrast this with the Cayley table representation where membership testing can be done in constant time since the elements are known and are already used as row and column indices of the Cayley table. In the Cayley representation the user knows the labels or the names of each group element explicitly and has a direct access to each element. The labels of the elements are often taken to be $1, 2, \ldots, n$ where n is the order of the group. The situation is quite different for permutation group representation, polycyclic presentation or generator-relator presentation. In these cases the user does not have an explicit representation for each element.

2 Preliminary

In this section we recall some definitions and notations which we use in this paper. In this paper we only consider finite groups. The number of elements in a group G is called the *order of* G and is denoted by $|G|$. A group G is abelian if $g_1g_2 = g_2g_1$ for all $g_1, g_2 \in G$. For a subgroup H of G and $g \in G$, the set $gH = \{gh \,|\, h \in H\}$ is called a left coset. Similarly, we can define right coset of G. The number of the left (or right) cosets of H in G is called the *index of* H *in* G and is denoted by $[G : H]$. A *left traversal* of H in G is a set containing exactly one element from each left coset and similarly we can define right traversals. The size of left (right) traversal is the same as the index $[G : H]$. For $g \in G$, the set $gHg^{-1} = \{gag^{-1} \,|\, a \in H\}$ is called a conjugate of the subgroup H. A subgroup H of G is said to be *normal* in G (denoted $H \trianglelefteq G$) if $gHg^{-1} = H$ for all $g \in G$. We define the *normalizer* of H in G to be the set $N_G(H) = \{g \in G \,|\, gHg^{-1} = H\}$. Note that, $N_G(H)$ is the largest subgroup in G in which H is normal.

A group G is called *simple* if G has no nontrivial normal subgroup. The Classification Theorem of Finite Simple Groups states that all the finite simple groups can be classified into the following five classes: (1) cyclic groups of prime order, (2) alternating groups, (3) classical groups, (4) exceptional groups of Lie type and (5) 26 sporadic simple groups.

We list all the classes of the finite simple groups later in the Classification Theorem for Simple Groups in Section 5. If G is a finite simple group of Lie-type over $\mathbb{F}_q$ where q is a power of some prime p, the Borel subgroup B of G is defined as the semidirect product of the Sylow p-subgroup of G with the maximal split torus T. The Borel subgroup is also the normalizer of the Sylow p-subgroup of the finite simple group (see [6], [27]).

For the purpose of this paper it might be sufficient to know some results on the *orders* of certain subgroups of simple groups. The reader may choose to skip the details of the structure of these groups. We indicate what kind of subgroups we are interested in and the results regarding the order of those subgroups as and when required. An interested reader may refer to the books by Carter [6], Wilson [27], or Aschbacher [3] for more details.

▶ **Definition 1** (see e.g., [10]). *A* subnormal series *of a group G is chain of subgroups*

$$1 = G_k \leq G_{k-1} \leq \cdots \leq G_1 \leq G_0 = G$$

such that $G_i \trianglelefteq G_{i-1}$, for all i.

▶ **Definition 2** (see e.g., [10]). *In a group G a sequence of subgroups*

$$1 = G_k \leq G_{k-1} \leq \cdots \leq G_1 \leq G_0 = G$$

is called a composition series *if $G_i \trianglelefteq G_{i-1}$ and G_{i-1}/G_i is simple for all $i \in [k]$. Here, k is the* composition length *of G.*

▶ **Theorem 3** (Jordan-Hölder Theorem see e.g., [10]). *Let G be a finite group with $G \neq 1$. Then*

(i) *G has a composition series.*

(ii) *The composition factors in a composition series are unique, namely, if $1 = N_r \leq N_{r-1} \leq \cdots \leq N_1 \leq N_0 = G$ and $1 = M_s \leq M_{s-1} \leq \cdots \leq M_1 \leq M_0 = G$ are two composition series for G, then $r = s$ and there is some permutation π of $\{1, 2, \ldots, r\}$ such that,*

$$\frac{M_{\pi(i)}}{M_{\pi(i)+1}} \cong \frac{N_i}{N_{i+1}}, for 1 \leq i \leq r.$$

▶ **Theorem 4** (Correspondence Theorem see e.g., [21]). *Let $K \trianglelefteq G$ and let $v : G \longrightarrow G/K$ be the canonical map i.e. $v(g) = Kg$ for all g. Then $S \mapsto v(S) = S/K$ is a bijection from the family of all those subgroups S of G which contain K to the family of all the subgroups of G/K.*

Model of computation. In this paper, we use an abstract model of computation known as the word-RAM model. In this model, data is stored in resisters and memory units. Each memory unit and resister can store $O(\log n)$ bits where n is the size of the input. The unit of storage is called *word.* The machine in the word-RAM model can access a word and do the usual arithmetic, logical, and comparison operations in constant time. The input size for our purpose is the order of the group. Without loss of generality, we can assume that the elements of groups are $1, 2, 3, ..., n$. Thus, every group element can be stored in a word and can be accessed in constant time.

There are two phases in the construction of a data structure: *the preprocessing phase* and *the query phase*. In the preprocessing phase, we assume that we have been given a finite group by its Cayley table. Using the Cayley table, we construct a data structure that consists of some arrays and tables. In the query phase, we process multiplication queries. In a multiplication query, two group elements g_1 and g_2 are given by the user. The task is to find the product of g_1 and g_2. In this phase, the data structure constructed in the preprocessing phase is accessed to answer the query. The time taken to answer a single query is called the *query-time*.

The time and space used in preprocessing stage are not considered. We only consider the space used by the data structure and the time it takes to answer a query to multiply the group elements.

▶ **Definition 5.** *Let G be a group and s and t be two positive real numbers. We say that G has an (s, t)-data structure, if G can be stored in a data-structure that uses at most s space and can answer a multiplication query in time at most t.*

▶ **Definition 6.** *Let $\mathcal{G}$ be a class of group and let $s, t : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be two functions. If for every group $G \in \mathcal{G}$ of order n there is a data structure that uses $O(s(n))$ space to store G and can answer a multiplication query in time at most $O(t(n))$ then we say that $\mathcal{G}$ has an $(O(s(n)), O(t(n)))$-data structure.*

3 Extension Theorems

In this section, we discuss how to use data structures for subgroups to build new data structures for groups containing the subgroups.

▶ **Theorem 7.** *There exist positive constants c and d such that for any group G and a subgroup H of G if H has an (s, t)-data structure for some s and t then G has an $(s + c([G : H]^2 + |G|), 2t + d)$-data structure.*

Proof. First we fix a left traversal L and a right traversal R of H in G. Each $g \in G$ can be uniquely written as $g = hr$ where $h \in H$ and $r \in R$. Thus we can define functions $s_R : G \longrightarrow H$ and $c_R : G \longrightarrow R$ such that $g = s_R(g)c_R(g)$. Similarly we can define $c_L : G \longrightarrow L$ and $s_L : G \longrightarrow H$ such that $g = c_L(g)s_L(g)$. We can store these four functions in four arrays each of length $|G|$.

Suppose we need to find the product of g_1 and g_2. Note that,

$$g_1g_2 = c_L(g_1)s_L(g_1)s_R(g_2)c_R(g_2).$$

Since $s_L(g_1), s_R(g_2) \in H$, we can use the data structure for H to find $s_L(g_1)s_R(g_2)$ within time t. Let $h_1 = s_L(g_1)s_R(g_2)$. Therefore, we can write $g_1g_2 = c_L(g_1)h_1c_R(g_2)$.

Given $l \in L$ and $h \in H$, we know that there exist unique elements $h' \in H$ and $r \in R$ such that $lh = h'r$. Thus, we can define two functions $Flip_H : L \times H \longrightarrow H$ and $Flip_R : L \times H \longrightarrow R$ such that $lh = Flip_H(l, h)Flip_R(l, h)$. We can store $Flip_H$ and $Flip_R$ in two 2-dimensional arrays using space linear in $|H \times L| = |G|$. With the help of these functions, we can write

$$g_1g_2 = Flip_H(c_L(g_1), h_1)Flip_R(c_L(g_1), h_1)c_R(g_2) = h_2r_1r_2$$

where $h_2 = Flip_H(c_L(g_1), h_1)$, $r_1 = Flip_R(c_L(g_1), h_1)$ and $r_2 = c_R(g_2)$.

Again we use the fact that any element g of G can be uniquely written as $g = hr$ where $h \in H$ and $r \in R$ to define the functions $Cross_H : R \times R \longrightarrow H$ and $Cross_R : R \times R \longrightarrow R$ such that for all $r, r' \in R$ we have $rr' = Cross_H(r, r')Cross_R(r, r')$. Note that we can store these functions in two 2-dimensional arrays each requiring size linear in $|R \times R| = (|G|/|H|)^2$. With the help of these functions we can write

$$g_1g_2 = h_2Cross_H(r_1, r_2)Cross_R(r_1, r_2) = h_2h_3r_3$$

where $Cross_H(r_1, r_2) = h_3$ and $r_3 = Cross_R(r_1, r_2)$.

Again we can use the data structure for H to compute the product $h_4 = h_2h_3$ within time t. Thus $g_1g_2 = h_4r_3$. Finally, we define a function $Fuse : H \times R \longrightarrow G$ simply as $Fuse(h, r) = hr$ for all $h \in H$ and $r \in R$. Clearly, a 2-dimensional array to store $Fuse$ would take space linear in $|H \times R| = |G|$. Thus, to produce the final result we just return $g_1g_2 = Fuse(h_4, r_3)$.

All the functions except for $Cross_R$ and $Cross_H$ take space linear in $|G|$, while $Cross_R$ and $Cross_H$ take space linear in $(|G|/|H|)^2$. The data structure for H takes space at most s. Therefore, the total space required is linear in $|G| + (|G|/|H|)^2$. We note that each function defined in this proof is queried exactly once. Thus, the time to query all the nine functions is bounded by some constant d. Additionally, the time taken to query the data structure for H is at most $2t$. Therefore, we have the required data structure for G. ◀

An immediate corollary of the above theorem is the following.

▶ **Corollary 8.** *Let $0 < c_1 \leq c_2$ be two constants. Let $\mathcal{G}_{c_1,c_2}$ be the class of groups G that has a subgroup H with $c_1\sqrt{|G|} \leq |H| \leq c_2\sqrt{|G|}$. Then $\mathcal{G}_{c_1,c_2}$ has $(O(n), O(1))$ data-structures.*

Proof. The Cayley table for H takes size at most $c_2^2|G|$ and answers queries in constant time. Since $|G|/|H| \leq (1/c_1)\sqrt{|G|}$, we have $(|G|/|H|)^2 \leq (1/c_1)^2|G|$. Hence the result follows from Theorem 7. ◀

In the next theorem we show how to use the data-structure for a normal subgroup of a group to build a data structure for the group when the quotient group is cyclic.

▶ **Theorem 9.** *There are positive constants c and d such that for every group G and any normal subgroup N of G, if G/N is cyclic and N has an (s, t)-data structure for some s and t, then G has an $(s + c|G|, 2t + d)$-data structure.*

Proof. Since G/N is cyclic it is generated by an element g_0N where $g_0 \in G$. The cosets of N in G are $N, g_0N, g_0^2N, \ldots, g_0^{k-1}N$ where k is the order of the group G/N, i.e., $k = [G : N]$. Clearly, $k \leq |G|$. Let $S = \{0, 1, \ldots, k-1\}$.

The set $\{g_0^0, g_0^1, \ldots, g_0^{k-1}\}$ is a left as well as a right traversal of N in G. Hence any element g could be uniquely written as $g = g_0^r n = n'g_0^r$ for some $r \in S$ and $n, n' \in N$. This enables us to define functions $e : G \longrightarrow S$, $s_R : G \longrightarrow N$ and $s_L : G \longrightarrow N$ such that for all $g \in G$

$$g = g_0^{e(g)}s_R(g) = s_L(g)g_0^{e(g)}.$$

These three functions could be stored in arrays each having size $|G|$. To multiply g_1 and g_2 we first observe that $g_1g_2 = g_0^{e(g_1)}s_R(g_1)s_L(g_2)g_0^{e(g_2)}$. These expression could be obtained by querying each of the functions once. The product $n_1 = s_R(g_1)s_L(g_2)$ can be obtained using the data structure for N within query-time t. Thus $g_1g_2 = g_0^{\alpha}n_1g_0^{\beta}$, where $\alpha = e(g_1)$ and $\beta = e(g_2)$.

Next we define a function $\mathit{Flip} : N \times S \longrightarrow N$ with the property that for all $n \in N$ and $i \in S$, $ng_0^i = g_0^i \mathit{Flip}(n,i)$. In other words, $\mathit{Flip}(n,i)$ is just $g_0^{-i} n g_0^i$. This function can be stored in space linear in $|N \times S| = |G|$. Now we can write $g_1 g_2 = g_0^\alpha g_0^\beta \mathit{Flip}(n_1, \beta) = g_0^{\alpha+\beta} n_2$ where $n_2 = \mathit{Flip}(n_1, \beta)$.

Next we compute $g_0^\alpha g_0^\beta = g_0^{\alpha+\beta}$. Observe that $\alpha + \beta \in \{0, 1, \ldots, 2k-2\}$. We define two functions $\mathit{red}_e : \{0, 1, \ldots, 2k-2\} \longrightarrow S$ and $\mathit{red}_N : \{0, 1, \ldots, 2k-2\} \longrightarrow N$ such that $g_0^\ell = g_0^{\mathit{red}_e(\ell)} \mathit{red}_N(\ell)$ for all $\ell \in \{0, \ldots, 2k-2\}$. Note that for $\ell < k$, $\mathit{red}_e(\ell) = \ell$ and $\mathit{red}_N(\ell) = \mathit{id}$. These two functions can be stored using space linear in k. Since $k \leq |G|$, the space required is at most linear in G.

Therefore,

$$g_1 g_2 = g_0^{\alpha+\beta} n_2 = g_0^{\mathit{red}_e(\alpha+\beta)} \mathit{red}_N(\alpha+\beta) n_2.$$

As before the product n_3 of $\mathit{red}_N(\alpha+\beta)$ and n_2 can be found using the data structure for N. Let $\mathit{red}_e(\alpha+\beta) = \gamma$. Hence, $g_1 g_2 = g_0^\gamma n_3$.

We finally define a function $\mathit{Fuse} : S \times N \longrightarrow G$ as $\mathit{Fuse}(i,n) = g_0^i n$ for all $i \in S$ and $n \in N$. Clearly, the function Fuse can be stored using space linear in $|G|$.

The product $g_1 g_2$ is just $\mathit{Fuse}(\gamma, n_3)$.

Each function defined in this proof takes space linear in $|G|$ and the data structure for N takes space at most s. Each function is queried exactly once and the data structure for N is queried twice. This proves the theorem. ◀

4 Compact Data Structures for Finite Groups

Let G be a group of order n. Our goal is to design a constant query-time data structure for G of size linear in n. We first consider a composition series $1 = G_k \lhd \ldots G_1 \lhd G_0 = G$ of G. In case there is a subgroup G_i in the composition series with size within a constant factor of $\sqrt{n}$, we can apply Corollary 8 to obtain a $(O(n), O(1))$ data structure for G. Otherwise we consider the smallest subgroup G_i of order more than $\sqrt{n}$. Note that here $|G_{i+1}|$ is at most $\sqrt{n}$ and therefore G_{i+1} will have its Cayley table of size at most n. This Cayley table can be used to answer a multiplication query involving elements in G_{i+1} in constant time.

Now we consider the composition factor G_i/G_{i+1}. This quotient is a simple group. If this is an abelian group it must be cyclic (of prime order) and we can use Theorem 9 to get a data structure for G_i. Then an application of Theorem 7 with G and its subgroup G_i will give us the required data structure for G.

The nontrivial case is when G_i/G_{i+1} is nonabelian. This is where we use the Classification Theorem of Finite Simple Groups. The classification theorem allows us to split the nonabelian case into various subcases. In each of the subcases we show that we can insert two subgroups G_{i_2} and G_{i_1} such that $G_{i+1} < G_{i_2} < G_{i_1} < G_i$ in such a manner that the indices $[G_{i_2} : G_{i+1}]$, $[G_{i_1} : G_{i_2}]$ and $[G_i : G_{i_1}]$ are all "small". Since G_{i+1} already has a constant query-time data structure (namely its Cayley table) of size linear in n, this allows us to use Theorem 7 successively to the group and subgroup pairs (G_{i_2}, G_{i+1}), (G_{i_1}, G_{i_2}), and (G_i, G_{i_1}) to obtain a constant query-time data structure for G_i of size linear in n. Finally, another application of Theorem 7 with G and its subgroup G_i will give us the required data structure for G.

4.1 Solvable Finite Groups

In this subsection we consider the class $\mathcal{G}_{solv}$ of finite solvable groups. We do this case first before going to the general case for the class of all finite groups because it is independent of the Classification Theorem for Finite Simple Groups.

▶ **Theorem 10.** *The class $\mathcal{G}_{solv}$ has $(O(n), O(1))$ data-structures.*

Proof. Let G be a group and $1 = G_k \lhd \ldots G_1 \lhd G_0 = G$ be a composition series of G. Let $n = |G|$.

Case 1: There is i such that $\sqrt{n}/2 \leq |G_i| \leq \sqrt{n}$. We simply apply Corollary 8 to get the desired data structure.

Case 2: There is no i such that $\sqrt{n}/2 \leq |G_i| \leq \sqrt{n}$. Let i be the largest index such that $\sqrt{n} < |G_i|$. We will have $|G_{i+1}| < \sqrt{n}/2$. The Cayley table for G_{i+1} has at most $n/4$ entries. Since G is solvable G_i/G_{i+1} is cyclic of prime order. This allows us to use Theorem 9 to obtain a constant query-time data structure for G_i which is linear in n. Next we observe that $[G : G_i]$ is at most $\sqrt{n}$. If we apply Theorem 7 on G and its subgroup G_i we get the required data structure for G. ◀

4.2 The General Case

Before considering the case for general finite groups we need the following result for nonabelian simple groups.

▶ **Lemma 11.** *There are positive constants b_1 and b_2 such that for any nonabelian simple group H there exist subgroups H_1 and H_2 such that $1 \leq H_2 \leq H_1 \leq H$ and $|H_2| \leq \sqrt{|H|}$, $[H : H_1] \leq b_1\sqrt{|H|}$, and $[H_1 : H_2] \leq b_2\sqrt{|H|}$.*

Proof. The proof uses the Classification Theorem of Finite Simple Group (CFSG). The proof idea is given in Section 5 and the details are given in the Appendix. ◀

Next we prove the main theorem of the paper. We note that Case 2 in the proof of the following theorem can be viewed as a generalized version of the problem of designing linear space and constant query-time data structure for nonableian simple groups.

▶ **Theorem 12.** *The class $\mathcal{G}_{fin}$ of all finite groups has $(O(n), O(1))$ data structures.*

Proof. Let G be a group of order n. We start by considering a composition series $1 = G_k \lhd \ldots G_1 \lhd G_0 = G$ be a composition series of G.

Case 1: This is the case when there is i such that $\sqrt{n}/2 \leq |G_i| \leq \sqrt{n}$. This case is exactly similar to the case for solvable groups.

Case 2: As before in this case we assume that there is no composition series element G_i with order more that $\sqrt{n}/2$ but less than $\sqrt{n}$. Let i be the largest index such $\sqrt{n} < |G_i|$. We will then have $|G_{i+1}| < \sqrt{n}/2$. Clearly, the Cayley table of G_{i+1} will have at most $n/4$ entries. Since $[G : G_i] < \sqrt{n}$, by Theorem 7 it is enough to design constant query-time data structure for G_i of size linear in n. In the rest of the proof we therefore concentrate on designing a constant query-time data structure for G_i that uses $O(n)$ space.

If the composition factor G_i/G_{i+1} is abelian then we are again in the same situation as in the second case of solvable groups. Therefore we assume that G_i/G_{i+1} is nonabelian.

We apply Lemma 11 to $H = G_i/G_{i+1}$ to obtain subgroups H_1 and H_2 such that $1 \leq H_2 \leq H_1 \leq H = G_i/G_{i+1}$. By the correspondence theorem of groups, H_1 and H_2 will be of the form G_{i_1}/G_{i+1} and G_{i_2}/G_{i+1} respectively for some subgroups G_{i_1} and G_{i_2} such that $G_{i+1} \leq G_{i_2} \leq G_{i_1} \leq G_i$. From Lemma 11 we have $[H_1 : H_2] \leq b_2\sqrt{|H|}$. Since, $H_1 = G_{i_1}/G_{i+1}$ and $H_2 = G_{i_2}/G_{i+1}$, we have $[G_{i_1}/G_{i+1} : G_{i_2}/G_{i+1}] \leq b_2\sqrt{|G_i/G_{i+1}|} \leq b_2\sqrt{n}$.

Therefore, $[G_{i_1} : G_{i_2}] \leq b_2\sqrt{n}$. Similarly, $[G_i : G_{i_1}] \leq b_1\sqrt{n}$. Again from Lemma 11, we have $H_2 \leq \sqrt{|H|}$. This implies, $[G_{i_2} : G_{i+1}] \leq \sqrt{|G_i/G_{i+1}|} \leq \sqrt{n}$.

Since G_{i+1} has a Cayley table of size at most n and $[G_{i_2} : G_{i+1}] \leq \sqrt{n}$, we will have a constant query-time data structure for the subgroup G_{i_2} of size at most n by Theorem 7. Since $[G_{i_1} : G_{i_2}] \leq b_2\sqrt{n}$ and $[G_i : G_{i_1}] \leq b_1\sqrt{n}$, another two applications of Theorem 7 with the group and subgroup pairs (G_{i_1}, G_{i_2}) and (G_i, G_{i_1}) will give a data structure for G_i of size linear in n which can answer a multiplication query in constant time. ◀

We note that there exist polynomial time algorithms for finding a composition series [22] and checking if a composition factor is abelian [14]. First, we note that G_{i_1} and G_{i_2} can be found simply by a brute force approach. Therefore, we can actually *construct* the data structure for G in the above theorem. While obtaining a polynomial time algorithm to construct the data structure is not our main goal, we note that we can also construct the data structure in polynomial time. The proof of this involves careful use of existing results from group theory and algorithms for group theoretic problems.

5 Proof Sketch for Lemma 11

In this section we sketch the proof idea behind Lemma 11. We first state the Classification Theorem of Finite Simple Groups.

▶ **Theorem 13** ([27]). *(The Classification Theorem of Finite Simple Group)*
Every finite simple group is isomorphic to one of the following:

(i) *a cyclic group* C_p *of prime order* p*;*
(ii) *an alternating group* A_m, *for* $m \geq 5$*;*
(iii) *a classical group;*
  a. *linear:* $A_{\mathrm{m}}(q)(\textit{or } \mathrm{PSL}_{\mathrm{m}+1}(q)), \mathrm{m} \geq 1,$ *except* $\mathrm{PSL}_2(2)$ *and* $\mathrm{PSL}_2(3)$*;*
  b. *unitary:* ${}^2A_{\mathrm{m}}(q^2)(\textit{or}\,\mathrm{PSU}_{\mathrm{m}+1}(q))\,, \mathrm{m} \geq 2,$ *except* $\mathrm{PSU}_3(2)$*;*
  c. *symplectic:* $C_{\mathrm{m}}(q))(\textit{or } \mathrm{PS}_{\mathrm{p2m}}(q)), \mathrm{m} \geq 2$ *except* $\mathrm{PS}_{p_4}(2)$*;*
  d. *orthogonal:* $B_{\mathrm{m}}(q)(\textit{or } \mathrm{P}\Omega_{2\mathrm{m}+1}(q)), \mathrm{m} \geq 3, q$ *odd;*
  $D_{\mathrm{m}}(q)(\textit{or } \mathrm{P}\Omega^{+}_{2\mathrm{m}}(q)), \mathrm{m} \geq 4;$
  ${}^2D_{\mathrm{m}}(q^2)(\textit{or } \mathrm{P}\Omega^{-}_{2\mathrm{m}}(q))\,, \mathrm{m} \geq 4$
  where q *is a power* p^a *of some prime;*
(iv) *an exceptional group of Lie type:*

$$\mathrm{G_2(q), q \geq 3; F_4(q); E_6(q);^2 E_6(q);^3 D_4(q); E_7(q); E_8(q)} \ \textit{or}$$

where q *is a power* p^a *of some prime;*

$$\mathrm{{}^2B_2(2^{2m+1}), m \geq 1;^2 G_2(3^{2m+1}), m \geq 1;^2 F_4(2^{2m+1}), m \geq 1}$$

or the Tits group ${}^2F_4(2)^{'}$*;*
(v) *one of 26 sporadic simple groups:*
  a. *the five Mathieu groups* $\mathrm{M_{11}, M_{12}, M_{22}, M_{23}, M_{24}}$*;*
  b. *the seven Leech Lattice groups* $\mathrm{Co_1, Co_2, Co_3, McL, HS, Suz, J_2}$*;*
  c. *the three Fischer groups* $\mathrm{Fi_{22}, Fi_{23}, Fi^{'}_{24}}$;
  d. *the Monstrous groups* $\mathbb{M}, \mathbb{B}, \mathrm{Th, HN, He}$*;*
  e. *the six pariahs* $\mathrm{J_1, J_2, J_4, O'N, Ly, Ru}$.

The definition of each of the group classes mentioned in the above theorem can be found in the standard texts on CFSG (see e.g., [6], [27], [3]).

Since Lemma 11 is about nonabelian simple groups we need to consider cases (ii) to (v) in Theorem 13. We take each subcases under these cases and show that there are subgroups H_1 and H_2 satisfying the conditions of the lemma.

We note that the 26 sporadic simple groups listed in the case (v) are of constant sizes. Therefore, we can ignore these groups for the purpose of the proof by simply taking H_2 to be the identity subgroup and H_1 to be H. Of course if we do so we need to pick extremely large constant b_2 as some the sporadic simple groups are of huge sizes. Fortunately, there are known results on the groups listed under case (v) that helps us to keep the constants b_1 and b_2 under 5.

We handle the *Alternating group* (case (ii) of Theorem 13) case as follows. Notice that one can find $k \in \mathbb{Z}$ such that $\frac{k!}{2} \leq \sqrt{\frac{m!}{2}} < \frac{(k+1)!}{2}$, and $H_2 \cong A_k$ and $H_1 \cong A_{k+1}$. Then clearly, $|H_2|^2 = \left(\frac{k!}{2}\right)^2 \leq \frac{m!}{2}$. The inequality $\frac{m!}{2} < \left(\frac{(k+1)!}{2}\right)^2$ implies that $k > \frac{m}{2}$. The value of k can be computed easily. One can also check that, $\left(\frac{|H_1|}{|H_2|}\right)^2 = (k+1)^2 \leq \frac{m!}{2}$ and $\left(\frac{|H|}{|H_1|}\right)^2 < \frac{m!}{2}$.

For the remaining groups we use the following two methods for the choices of H_1 and H_2. The methods are as follows:

1. **Method 1:** In this method, we first choose H_2 to be a certain Sylow subgroup of the given simple group H. Next we pick H_1 to be the normalizer of H_2 in H or the Borel subgroup containing H_2..

   Example: Let us take H to be a simple group $A_m(q)$ for some $q > 2$ which appears in case (iii) of Theorem 13. Here q is power of some prime p. It is known that $A_m(q)$ has order $q^{m(m+1)/2} \prod_{i=1}^{m} (q^{i+1} - 1)/(q-1, m+1)$ where $(q-1, m+1)$ denotes the gcd of $q-1$ and $m+1$ (see [3], p. 252). Clearly, H will have a Sylow p-subgroup of order $q^{m(m+1)/2}$. We set H_2 to be this subgroup. Next we pick H_1 to be the normailzer of H_2 in H. It is also known that the order of H_1 is $q^{m(m+1)/2}(q-1)^m$ (see [27], p. 46). One can check that with $b_1 = 2$ and $b_2 = 1$, these choices satisfy the conditions of Lemma 11 (see Appendix for the details).

2. **Method 2:** In this method, we choose H_1 to be a maximal subgroup of the simple group H and H_2 to certain Sylow subgroup of H_1.

   Example: In the example under Method 1 we consider the case for $A_m(q)$ when $q > 2$. In this example we take the case when $q = 2$. Here $H = A_m(q)$ will have order $2^{m(m+1)/2} \prod_{i=1}^{m} (2^{i+1} - 1)$ (see [3], p. 252). It is known that the maximal subgroup of H is of order $|H|/(2^m - 1)$ (see [16], p. 175). We take this subgroup as H_1. Next we take H_2 as a Sylow 2-subgroup of H_1 which has order $2^{m(m+1)/2}$. It is easy to verify that these choices of H_1 and H_2 along with $b_1 = b_2 = 1$ satisfy the conditions of Lemma 11 (see Appendix for the details).

Table 1 lists the methods that we have used for choosing the suitable subgroups in the corresponding nonabelian simple group. The last two columns represent the constant factors b_1 and b_2 for the corresponding simple group (see Table 1).

For case (v), we use Method 2 to get the suitable subgroups.

Appendix A.3 contains two comprehensive tables listing the orders of subgroups used in the proof of Lemma 11 for different cases of CFSG.

Table 1 Table representing the constant factor and method used for choosing suitable subgroups.

Case	H	Condition on q	Method	b_1	b_2
(iii)	$A_m(q)$	$q > 2$	Method 1	2	1
		$q = 2$	Method 2	1	1
	${}^2A_m(q^2); m > 1$	$q > 2$	Method 1	2	1
		$q = 2; 6 \nmid (m-1)$	Method 2	1	1
		$q = 2; 6 \mid (m-1)$	Method 2	1	1
	$C_m(q); m > 2$	$q > 2$	Method 1	2	1
		$q = 2$	Method 2	1	1
	$B_m(q); m > 1$	q odd	Method 1	2	1
	$D_m(q); m > 3$	$q > 2$	Method 1	2	1
		$q = 2$	Method 2	1	1
	${}^2D_m(q^2); m > 3$	$q > 2$	Method 1	3	1
		$q = 2$	Method 2	1	1
(iv)	$G_2(q)$	$q \geq 3$	Method 1	1	1
	$F_4(q)$	All q	Method 2	1	1
	$E_6(q)$	$q > 2$	Method 1	1	1
		$q = 2$	Method 2	1	1
	${}^2E_6(q)$	All q	Method 1	1	1
	${}^3D_4(q)$	All q	Method 1	1	1
	$E_7(q)$	$q > 2$	Method 1	1	1
		$q = 2$	Method 2	1	1
	$E_8(q)$	$q > 2$	Method 1	1	1
		$q = 2$	Method 2	1	1
	${}^2B_2(q)$	$q = 2^{2t+1}, t \geq 1$	Method 1	1	1
	${}^2G_2(q)$	$q = 3^{2t+1}, t \geq 1$	Method 1	1	1
	${}^2F_4(q)$	$q = 2^{2t+1}, t \geq 1$	Method 1	1	1
	${}^2F_4(2)'$	$q = 2$	Method 2	1	1

References

1 Vikraman Arvind and Jacobo Torán. The complexity of quasigroup isomorphism and the minimum generating set problem. In *International Symposium on Algorithms and Computation*, pages 233–242. Springer, 2006.

2 Vikraman Arvind and Jacobo Torán. Solvable group isomorphism is (almost) in NP ∩ conp. *ACM Trans. Comput. Theory*, 2(2):4:1–4:22, 2011. doi:10.1145/1944857.1944859.

3 M. Aschbacher. *Finite Group Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2000. doi:10.1017/CBO9781139175319.

4 László Babai, Paolo Codenotti, and Youming Qiao. Polynomial-time isomorphism test for groups with no abelian normal subgroups. In *International Colloquium on Automata, Languages, and Programming*, pages 51–62. Springer, 2012.

5 László Babai and Youming Qiao. Polynomial-time isomorphism test for groups with abelian sylow towers. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 453–464. LIPIcs, 2012.

6 Roger W. Carter. *Finite groups of Lie type*. Wiley Classics Library. John Wiley & Sons, Ltd., Chichester, 1993. Conjugacy classes and complex characters, Reprint of the 1985 original, A Wiley-Interscience Publication.

7 J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. *ATLAS of Finite Groups.* Oxford University Press, Eynsham, 1985. Maximal subgroups and ordinary characters for simple groups, With computational assistance from J. G. Thackray.

8 Bireswar Das and Shivdutt Sharma. Compact data structures for dedekind groups and finite rings. In *WALCOM*, pages 90–102, 2021.

9 Bireswar Das, Shivdutt Sharma, and P. R. Vaidyanathan. Space efficient representations of finite groups. *J. Comput. Syst. Sci.*, 114:137–146, 2020. doi:10.1016/j.jcss.2020.06.007.

10 David S. Dummit and Richard M. Foote. *Abstract algebra.* John Wiley & Sons, Inc., Hoboken, NJ, third edition, 2004.

11 Arash Farzan and J. Ian Munro. Succinct representation of finite abelian groups. In *ISSAC 2006*, pages 87–92. ACM, New York, 2006.

12 Merrick Furst, John Hopcroft, and Eugene Luks. Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 36–41. IEEE, 1980.

13 François Le Gall. Efficient isomorphism testing for a class of group extensions. *CoRR*, abs/0812.2298, 2008. arXiv:0812.2298.

14 T. Kavitha. Linear time algorithms for abelian group isomorphism and related problems. *J. Comput. System Sci.*, 73(6):986–996, 2007.

15 Neeraj Kayal and Timur Nezhmetdinov. Factoring groups efficiently. In *International colloquium on automata, languages, and programming*, pages 585–596. Springer, 2009.

16 Peter B. Kleidman and Martin W. Liebeck. *The Subgroup Structure of the Finite Classical Groups.* London Mathematical Society Lecture Note Series. Cambridge University Press, 1990. doi:10.1017/CBO9780511629235.

17 Daniel J. Kleitman, Bruce R. Rothschild, and Joel H. Spencer. The number of semigroups of order n. *Proc. Amer. Math. Soc.*, 55(1):227–232, 1976.

18 S Ravi Kumar and Ronitt Rubinfeld. Property testing of abelian group operations, 1998.

19 Gary L. Miller. On the $n^{\log n}$ isomorphism technique: A preliminary report. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 51–58. ACM, 1978.

20 Youming Qiao, Jayalal Sarma, and Bangsheng Tang. On isomorphism testing of groups with normal hall subgroups. *J. Comput. Sci. Technol.*, 27(4):687–701, 2012. doi:10.1007/s11390-012-1255-7.

21 Joseph J. Rotman. *An introduction to the theory of groups*, volume 148 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, fourth edition, 1995.

22 Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics.* Cambridge University Press, Cambridge, 2003.

23 Charles C. Sims. Computational methods in the study of permutation groups. In John Leech, editor, *Computational Problems in Abstract Algebra*, pages 169–183. Pergamon, 1970. doi:10.1016/B978-0-08-012975-4.50020-5.

24 Charles C Sims. Computation with permutation groups. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, pages 23–28, 1971.

25 Charles C. Sims. *Computation with finitely presented groups*, volume 48 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 1994.

26 J. H. van Lint and R. M. Wilson. *A course in combinatorics.* Cambridge University Press, Cambridge, 1992.

27 Robert A. Wilson. *The finite simple groups*, volume 251 of *Graduate Texts in Mathematics.* Springer-Verlag London, Ltd., London, 2009. doi:10.1007/978-1-84800-988-2.

28 Robert A. Wilson. Maximal subgroups of sporadic groups. In *Finite simple groups: thirty years of the atlas and beyond*, volume 694 of *Contemp. Math.*, pages 57–72. Amer. Math. Soc., Providence, RI, 2017.

A Proof of Lemma 11

In this section we indicate how to prove Lemma 11 in more detail. We do this in the ordering mentioned in the Classification Theorem of Finite Simple Group, i.e., Theorem 13. As we mentioned in Section 2, we just need to use some known results on the *order* of certain subgroups of simple groups. The detailed description of these groups may be skipped for the purpose of the proof. The results that are used in the proof are on the orders of the finite simple groups, on the orders of maximal subgroups of simple groups and the normalizers of certain types of Sylow subgroups of simple groups. The information about the order of these simple groups can be obtained from [3]

In case (ii) of Theorem 13, H is an alternating group. We have already seen that the subgroups H_2 and H_1 are certain suitably picked stabilizer subgroups of the given alternating group H.

For the cases (iii) and (iv) of Theorem 13, we use Method 1 and Method 2 to get the desired subgroups as required in Lemma 11. In this cases the finite simple group H is of Lie-type and is defined over a finite field $\mathbb{F}_q$ where q is a power of some prime p. In Method 1, we take H_2 to be certain Sylow p-subgroup of H. The existence of such H_2 follows from the well-known Sylow theorem. For the existence of H_1, we take the normalizer of H_2 or the Borel subgroup. The information about the order of normalizer has been obtained from (see [6], p. 76, [27], p. 46).

For the groups in which we use Method 2, we consider a maximal subgroup of H as H_1 and H_2 to be some Sylow p-subgroup of H_1. The index of a maximal subgroup (and hence its order) can be obtained from [16], p. 175 and [27], p. 156.

For the simple groups in case (v), we use Method 2 and the information about order of maximal subgroup (H_1) can be obtained from [28]. Also, for the choice of H_2, we choose certain Sylow subgroup of H_1.

The inequalities in the following two remarks are used in the calculation multiple times.

▶ Remark 14. For all integer $q > 2$, we have $\frac{q}{(q-1)^2} < 1$.

▶ Remark 15. $\prod_{i=1}^{i=m}(q^{i+1} - (-1)^{i+1}) < q^{\sum_{i=1}^{i=m}(i+1)}$.

▶ Remark 16. The gcd of two natural numbers m and n is denoted by (m, n).

A.1 The Classical Groups

We have seen the case(ii) of Theorem 13 in Section 5. In this section, we consider H to be a classical simple group described in case (iii) of Theorem 13. In particular, we consider the case when H is ${}^2A_m(q^2)$ where q is a power of some prime p. All the other cases can be handle similarly. As described earlier, we use Method 1 and Method 2 to show the existence of subgroups H_2 and H_1 of the simple group H.

1.1 $H = {}^2A_m(q^2)$; $m \geq 2$, $q > 2$ (Method 1)

The finite simple group ${}^2A_m(q^2)$ is isomorphic to the *projective special unitary group* $\mathrm{PSU}_{m+1}(q)$. The group $\mathrm{PSU}_{m+1}(q)$ is the group obtain by taking special unitary group $\mathrm{SU}_{m+1}(q)$ and quotienting it by its center, i.e. ${}^2A_m(q^2) \cong \frac{\mathrm{SU}_{m+1}(q)}{Z(\mathrm{SU_{m+1}(q)})}$ (see [27], p. 66). It is known that (see [3], p. 252) the order of ${}^2A_m(q^2)$ is,

$$|H| = \frac{q^{\frac{m(m+1)}{2}}}{(q+1, m+1)} \prod_{i=1}^{m}(q^{i+1} - (-1)^{i+1}).$$

Let H_2 be the Sylow p-subgroup of ${}^2A_m(q^2)$, then $|H_2| = q^{\frac{m(m+1)}{2}}$ and $|H_2|^2 \leq |H|$. Let H_1 be the Borel subgroup of H of order (see [27], [6]),

$$|H_1| = \frac{q^{\frac{m(m+1)}{2}}}{(q+1,m+1)}(q-1)^{\lfloor m/2 \rfloor}(q+1)^{\lceil \frac{m-1}{2} \rceil}.$$

One can check that $\left(\frac{|H_1|}{|H_2|}\right)^2 \leq |H_1| < |H|$ and $\frac{|H|}{|H_1|} \leq 2\sqrt{|H|}$.

1.2 $H = {}^2A_m(q^2)$; $m \geq 2$, $q = 2$ (Method 2)

The finite simple group ${}^2A_m(2^2)$ is of order $2^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(2^{i+1} - (-1)^{i+1})/(3, m+1)$ and is isomorphic to *projective special unitary group* $\mathrm{PSU}_{m+1}(2)$ or $\mathrm{U}_{m+1}(q)$. The group $\mathrm{U}_{m+1}(q)$ has a maximal subgroup of index $\frac{(2^{m+1}-(-1)^{m+1})(2^m-(-1)^m)}{3}$ when $6 \nmid (m-1)$ and of index $\frac{2^m(2^{m+1}-1)}{3}$, when $6 \mid (m-1)$ (see [16], p. 175) .

(Case 1) $6 \nmid (m-1)$

Let H_1 be corresponding maximal subgroup of ${}^2A_m(2^2)$ whose index is

$$\frac{(2^{m+1} - (-1)^{m+1})(2^m - (-1)^m)}{3}$$

in ${}^2A_m(2^2)$. Then, the order of H_1 is,

$$|H_1| = \frac{3}{(3, m+1)} \frac{2^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(2^{i+1} - (-1)^{i+1})}{(2^{m+1} - (-1)^{m+1})(2^m - (-1)^m)}.$$

Let H_2 be the Sylow 2-subgroup of H_1. Then, $|H_2| = 2^{\frac{m(m+1)}{2}}$ and $|H_2|^2 < |\,{}^2A_m(2^2)|$.

It is easy to see that $\frac{\left(\frac{|H_1|}{|H_2|}\right)^2}{|\,{}^2A_m(2^2)|} < 1$ and $\left(\frac{|\,{}^2A_m(2^2)|}{|H_1|}\right)^2 < |\,{}^2A_m(2^2)|$.

(Case 2) $6|(m-1)$ (i.e. $m \geq 7$)

In this case, as we know that the group ${}^2A_m(q^2)$ has a maximal subgroup of index $\frac{2^m(2^{m+1}-1)}{3}$. Let H_1 be one such maximal subgroup. Then,

$$|H_1| = \frac{3}{(3, m+1)} 2^{\frac{m(m-1)}{2}} \prod_{i=1}^{m-1}(2^{i+1} - (-1)^{i+1}).$$

Let H_2 be the Sylow 2-subgroup of H_1, then H_2 has order $2^{\frac{m(m-1)}{2}}$ and $|H_2|^2 < |\,{}^2A_m(2^2)|$.

Clearly we can check that $\left(\frac{|H_1|}{|H_2|}\right)^2 < |\,{}^2A_m(2^2)|$ and $\left(\frac{|\,{}^2A_m(2^2)|}{|H_1|}\right)^2 < |\,{}^2A_m(2^2)|$.

A.2 Exceptional Group of Lie Type

In this section, we consider H to be an exceptional simple group of Lie Type described in case (iv) of Theorem 13. In particular, we consider the case when H is $F_4(q)$, $E_6(q)$ and ${}^2F_4(2)'$ where q is a power of some prime p. The similar arguments can be used to prove the remaining cases.

(1) $H = F_4(q)$ (Method 2)

The finite simple group $F_4(q)$ has order (see [3], p. 252),

$$|F_4(q)| = q^{24}(q^{12} - 1)(q^8 - 1)(q^6 - 1)(q^2 - 1).$$

It is known that (see [27], p. 156) the group $F_4(q)$ has a maximal subgroup $q^{1+14} : Sp_6(q).C_{q-1}$ of order $q^{24}(q^6-1)(q^4-1)(q^2-1)(q-1)$ say H_1. This subgroup has a Sylow p-subgroup say H_2 of order q^{24} and $|H_2|^2 \leq |F_4(q)|$. Therefore, $\left(\frac{|H_1|}{|H_2|}\right)^2 < |H_1| < |H|$ and $\left(\frac{|H|}{|H_1|}\right)^2 < |H|$.

(2) $H = E_6(q)$; $q > 2$ (Method 1)
The group $E_6(q)$ is a finite simple group. The order of $H = E_6(q)$ is (see [3], p. 252),

$$|E_6(q)| = \frac{q^{36}}{(3, q-1)}(q^{12}-1)(q^9-1)(q^8-1)(q^6-1)(q^5-1)(q^2-1).$$

Clearly, it has a Sylow p-subgroup H_2 of order q^{36} and $|H_2|^2 \leq |E_6(q)|$. Let H_1 be the Borel subgroup of H then the order of H_1 is $q^{36}(q-1)^6$ (see [27], [6]). Clearly, $\left(\frac{|H_1|}{|H_2|}\right)^2 < |H_1| < |H|$ and $\left(\frac{|H|}{|H_1|}\right)^2 \leq |H|$.
Notice that, the group $E_6(2)$ is of constant order. However, we can use Method 2 to reduce the constants b_1, b_2 to 1. By taking H_1 to be maximal subgroup of order (see [7]) $2^{36} \cdot 3^3 \cdot 5 \cdot 7 \cdot 31$ and H_2 to be its Sylow 2-subgroup of order 2^{36}.

(3) $H = {}^2F_4(2)'$; (Method 2)
The simple group $H = {}^2F_4(2)'$; has order 17971200. It is known that H has a maximal subgroup of order 11232. We take H_1 to be this maximal subgroup and H_2 to be the Sylow 2-subgroup of H_1 which has order 32. Thus, we get $b_1 = b_2 = 1$.

A.3 Tables

In this section we cover the details of Sporadic simple groups (Table 2), and the order of all the simple groups that we define in cases (ii)-(iv) of Theorem 13 in Table 3 and 4. These tables also contain the order of the subgroups H_2 and H_1.
Table 2 represents the information about the subgroups H_2 and H_1 of the Sporadic simple groups. In Table 2 we consider the values of $t_i, i = 1, 2, 3, 4$ as follows.

$t_1 = 808017424794512875886459904961710757005754368000000000$
$t_2 = 2^{42} \cdot 3^{13} \cdot 5^6 \cdot 7^2 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 31 \cdot 47$
$t_3 = 4154781481226426191177580544000000$
$t_4 = 2^{38} \cdot (2^{12}-1) \cdot (2^9+1) \cdot (2^8-1) \cdot (2^6-1) \cdot (2^5+1) \cdot (2^2-1)$

In the Table 3, the values of $c_{mi}, i = 1, 2, 3, 4, 5$ are as follows.

$c_{m1} = \frac{q^{\frac{m(m+1)}{2}}}{(q+1,m+1)}\ (q-1)^{\lfloor m/2 \rfloor}(q+1)^{\lceil \frac{m-1}{2} \rceil}.$
$c_{m2} = \frac{3}{(3,m+1)} \frac{2^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(2^{i+1}-(-1)^{i+1})}{(2^{m+1}-(-1)^{m+1})(2^m-(-1)^m)}$
$c_{m3} = \frac{3}{(3,m+1)} 2^{\frac{m(m-1)}{2}} \prod_{i=1}^{m-1}(2^{i+1}-(-1)^{i+1})$
$c_{m4} = 2^{m^2-m+1}(2^m+1)\prod_{i=1}^{m-1}(2^{2i}-1)$
$c_{m5} = 2^{m(m-1)}(2^{m-1}+1)\prod_{i=1}^{m-2}(2^{2i}-1).$

Table 2 Table representing the constant factor and Method used for choosing suitable subgroups.

H	Order of H	Order of H_2	Order of H_1	b_1	b_2
M_{11}	7920	2^4	720	1	1
M_{12}	95040	2^2	660	1	1
M_{22}	443520	2^6	20160	1	1
M_{23}	10200960	2^7	443520	1	1
M_{24}	244823040	2^8	887040	1	1
Co_1	4157776806543360000	262144	42305400000000	1	1
Co_2	42305400000000	262144	908328960	1	1
Co_3	495767000000	2^7	10200960	1	1
McL	898128000	3^6	$3^6 \cdot 2^7 \cdot 7 \cdot 5$	1	1
HS	44352000	2^7	$2^7 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11$	1	1
Suz	448345497600	2^{12}	251596800	1	1
J_2	604800	2^5	6048	1	1
Fi_{22}	64561751654400	2^{16}	$2^{16}(2^6-1)(2^5+1)(2^4-1)(2^3+1)$	1	1
Fi_{23}	4089470473293004800	2^{18}	$2^{18} \cdot 3^9 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$	1	1
$\mathrm{Fi}^{`}_{24}$	1255205709190661721292800	2^{19}	$2^{19} \cdot 3^{13} \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 23$	1	1
$\mathbb{M}$	t_1	2^{42}	t_2	1	1
$\mathbb{B}$	t_3	2^{38}	t_4	1	1
Th	90745943887872000	2^{15}	319979520	1	1
HN	273030912000000	2^9	239500800	1	1
He	4030387200	2^8	$2^8 \cdot 255 \cdot 15$	1	1
J_1	175560	2^2	660	1	1
J_3	50232960	2^5	8160	1	1
J_4	86775571046077562880	2097152	57161637225	1	1
O′N	460815505920	2^6	3753792	1	1
Ly	51765179004000000	15625	5859000000	1	5
Ru	145926144000	2^{12}	35942400	1	1

Table 3 Order of the simple groups (case (iii) of Theorem 13) and order of its subgroups H_2, H_1.

H	$\lvert H\rvert$	$\lvert H_2\rvert$	$\lvert H_1\rvert$
$A_m(q)$; $q > 2$	$\frac{q^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(q^{i+1}-1)}{(q-1,m+1)}$	$q^{\frac{m(m+1)}{2}}$	$\frac{q^{\frac{m(m+1)}{2}}}{(q-1,m+1)}(q-1)^m$
$A_m(2)$; $q = 2$	$2^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(2^{i+1}-1)$	$2^{\frac{m(m+1)}{2}}$	$\frac{2^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(2^{i+1}-1)}{(2^{m+1}-1)}$
${}^2A_m(q^2)$; $q > 2$, $m > 1$	$\frac{q^{\frac{m(m+1)}{2}}}{(q+1,m+1)} \prod_{i=1}^{m}(q^{i+1}-(-1)^{i+1})$	$q^{\frac{m(m+1)}{2}}$	c_{m1}
${}^2A_m(2^2)$; $q = 2$, $m > 1$, $6 \nmid (m-1)$	$\frac{2^{\frac{m(m+1)}{2}}}{(3,m+1)} \prod_{i=1}^{m}(2^{i+1}-(-1)^{i+1})$	$2^{\frac{m(m+1)}{2}}$	c_{m2}
${}^2A_m(2^2)$; $q = 2$, $m > 1$, $6 \mid (m-1)$	$\frac{2^{\frac{m(m+1)}{2}}}{(3,m+1)} \prod_{i=1}^{m}(2^{i+1}-(-1)^{i+1})$	$2^{\frac{m(m-1)}{2}}$	c_{m3}
$C_m(q)$; $q > 2$, $m > 2$	$\frac{q^{m^2} \prod_{i=1}^{m}(q^{2i}-1)}{(2,q-1)}$	q^{m^2}	$\frac{q^{m^2}}{(2,q-1)}(q-1)^m$
$C_m(2)$; $q = 2$, $m > 2$	$2^{m^2} \prod_{i=1}^{m}(2^{2i}-1)$	2^{m^2-m+1}	c_{m4}
$B_m(q)$; q odd, $m > 1$	$\frac{q^{m^2} \prod_{i=1}^{m}(q^{2i}-1)}{(2,q-1)}$	q^{m^2}	$\frac{q^{m^2}}{(2,q-1)}(q-1)^m$
$D_m(q)$; $q > 2$, $m > 3$	$\frac{q^{m(m-1)}(q^m-1) \prod_{i=1}^{m-1}(q^{2i}-1)}{(4,q^m-1)}$	$q^{m(m-1)}$	$\frac{q^{m(m-1)}}{(4,q^m-1)}(q-1)^m$
$D_m(2)$; $q = 2$, $m > 3$	$2^{m(m-1)}(2^m-1) \prod_{i=1}^{m-1}(2^{2i}-1)$	2^{m^2-2m+1}	$2^{m^2-2m+1} \prod_{i=1}^{m-1}(2^{2i}-1)$
${}^2D_m(q^2)$; $q > 2$, $m > 3$	$\frac{q^{m(m-1)}(q^m+1)}{(4,q^m+1)} \prod_{i=1}^{m-1}(q^{2i}-1)$	$q^{m(m-1)}$	$\frac{q^{m(m-1)}}{(4,q^n+1)}(q-1)^m$
${}^2D_m(2^2)$; $q = 2$, $m > 3$	$\frac{2^{m(m-1)}(2^m+1)}{(4,2^m+1)} \prod_{i=1}^{m-1}(2^{2i}-1)$	$2^{m(m-1)}$	c_{m5}

Table 4 Order of the simple groups (case (iv) of Theorem 13) and order of its subgroups H_2, H_1.

$G_2(q)$	$q^6(q^6-1)(q^2-1)$	q^6	$q^6(q-1)^2$
$F_4(q)$	$q^{24} \prod_{i\in\{2,6,8,12\}}(q^i-1)$	q^{24}	$q^{24} \prod_{i\in\{1,2,4,6\}}(q^i-1)$
$E_6(q)$, $q > 2$	$\frac{q^{36}}{(3,q-1)} \prod_{i\in\{2,5,6,8,9,12\}}(q^i-1)$	q^{36}	$q^{36}(q-1)^6$
$E_6(2)$	$2^{36} \prod_{i\in\{2,5,6,8,9,12\}}(2^i-1)$	2^{36}	$2^{36} \cdot 3^3 \cdot 5 \cdot 7 \cdot 31$
${}^2E_6(q)$	$\frac{q^{36}(q^9+1)}{(3,q+1)} \prod_{i\in\{2,5,6,8,12\}}(q^i-1)$	q^{36}	$q^{36}(q-1)^4(q+1)^2$
${}^3D_4(q)$	$q^{12}(q^8+q^4+1)(q^6-1)(q^2-1)$	q^{12}	$q^{12}\ (q^3-1)(q-1)$
$E_7(q)$, $q > 2$	$\frac{q^{63}}{(2,q-1)} \prod_{i\in\{2,6,8,10,12,14,18\}}(q^i-1)$	q^{63}	$q^{63}(q-1)^7$
$E_7(2)$	$2^{63} \prod_{i\in\{2,6,8,10,12,14,18\}}(2^i-1)$	2^{63}	$2^{63} \cdot 3^4 \cdot 7^2 \cdot 5$
$E_8(q)$, $q > 2$	$q^{120} \prod_{i\in\{2,8,12,14,18,20,24,30\}}(q^i-1)$	q^{120}	$q^{120}(q-1)^8$
$E_8(2)$	$2^{120} \prod_{i\in\{2,8,12,14,18,20,24,30\}}(2^i-1)$	2^{119}	$2^{119} \cdot 3^4 \cdot 5 \cdot 7^2 \cdot 31$
${}^2B_2(q)$; $q = 2^{2t+1}$, $t \geq 1$	$q^2(q^2+1)(q-1)$	q^2	$q^2(q-1)$
${}^2G_2(q)$; $q = 3^{2t+1}$, $t \geq 1$	$q^3(q^3+1)(q-1)$	q^3	$q^3(q-1)$
${}^2F_4(q)$; $q = 2^{2t+1}$, $t \geq 1$	$q^{12}(q^6+1)(q^4-1)(q^3+1)(q-1)$	q^{12}	$q^{12}(q-1)^2$

The Isomorphism Problem for Plain Groups Is in Σ_3^{P}

Heiko Dietrich
Monash University, Clayton, Australia

Murray Elder[1]
University of Technology Sydney, Ultimo, Australia

Adam Piggott
Australian National University, Canberra, Australia

Youming Qiao
University of Technology Sydney, Ultimo, Australia

Armin Weiß
Universität Stuttgart, Germany

Abstract

Testing isomorphism of infinite groups is a classical topic, but from the complexity theory viewpoint, few results are known. Sénizergues and the fifth author (ICALP2018) proved that the isomorphism problem for virtually free groups is decidable in PSPACE when the input is given in terms of so-called virtually free presentations. Here we consider the isomorphism problem for the class of *plain groups*, that is, groups that are isomorphic to a free product of finitely many finite groups and finitely many copies of the infinite cyclic group. Every plain group is naturally and efficiently presented via an inverse-closed finite convergent length-reducing rewriting system. We prove that the isomorphism problem for plain groups given in this form lies in the polynomial time hierarchy, more precisely, in Σ_3^{P}. This result is achieved by combining new geometric and algebraic characterisations of groups presented by inverse-closed finite convergent length-reducing rewriting systems developed in recent work of the second and third authors (2021) with classical finite group isomorphism results of Babai and Szemerédi (1984).

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Rewrite systems; Mathematics of computing → Discrete mathematics; Theory of computation → Computability

Keywords and phrases plain group, isomorphism problem, polynomial hierarchy, Σ_3^{P} complexity class, inverse-closed finite convergent length-reducing rewriting system

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.26

Funding *Heiko Dietrich*: Supported by Australian Research Council grant DP190100317.
Murray Elder: Supported by Australian Research Council grant DP210100271.
Adam Piggott: Supported by Australian Research Council grant DP210100271.
Youming Qiao: Supported by Australian Research Council grant DP200100950.

Acknowledgements We wish to thank the reviewers for their helpful comments and corrections.

1 Introduction

The classical core of combinatorial group theory centres on Dehn's three algorithmic problems concerning finitely presented groups [7]: given a finite presentation for a group, describe an algorithm that decides whether or not an arbitrary word in the generators and their

[1] Corresponding author

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 26; pp. 26:1–26:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

inverses spells the identity element (the word problem); given a finite presentation for a group, describe an algorithm that decides whether or not two arbitrary words in the generators and their inverses spell conjugate elements (the conjugacy problem); describe an algorithm that, given two finite presentations, decides whether or not the groups presented are isomorphic (the isomorphism problem). For arbitrary finite presentations, these problems are undecidable and so upper bounds on complexity are impossible. Obtaining bounds on complexity requires working with presentations that come with a promise that they determine groups in a particular class, or presentations that provide (intrinsic or extrinsic) additional structure. From Dehn's work, finite length-reducing rewriting systems that satisfy various convergence properties emerged as finite group presentations that simultaneously specify interesting infinite groups and provide natural solutions to the corresponding word problems.

A research program, active since the 1980s, seeks to classify the groups that may be presented by finite length-reducing rewriting systems satisfying various convergence properties. The hyperbolic groups [15], the virtually-free groups [14] (groups with a free subgroup of finite index – by Muller and Schupp's theorem [23] they are also known as context-free groups), and the plain groups [10] are important classes of groups, each a proper subclass of the class before, that arise within this program. A group is *plain* if it is isomorphic to a free product of finitely many finite groups and a free group of finite rank. The plain groups may be characterised as the fundamental groups of finite graphs of finite groups with trivial edge groups [18], and as the groups admitting a finite group presentation with a simple reduced word problem [16]. Moreover, the plain groups are conjectured to be exactly the groups that may be presented by finite convergent length-reducing rewriting systems [21].

The isomorphism problem is, of course, the most difficult of Dehn's problems and complexity results concerning this problem are rare. However, progress has been made on the isomorphism problem for the very classes of groups that arise in the study of length-reducing rewriting systems. Krstić solved the isomorphism problem for virtually-free groups described by arbitrary finite group presentations [19]. Building on the pioneering work of Rips and Sela [26], Sela [27], and Dahmani and Groves [5], Dahmani and Guirardel [6] provided an explicit algorithm that solves the isomorphism problem in all hyperbolic groups when the groups are given by finite presentations. In light of this result, attention can now shift to complexity bounds for the isomorphism problem. Notice that, in order to obtain complexity bounds, we cannot allow arbitrary presentations as inputs (otherwise we could decide within that complexity bound whether a given presentation is for the trivial group – a problem which is undecidable). In [28, 29], Sénizergues showed that the isomorphism problem for virtually-free groups is primitive recursive when the input is given in the form of two virtually-free presentations, or as two context free grammars. A virtually-free presentation of a group G specifies a free subgroup F plus a set of representatives S for the cosets $F \setminus G$ together with relations describing pairwise multiplications of elements from F and S; a context-free grammar can specify a virtually-free group by generating the language of words that spell the identity element. Then in 2018, Sénizergues and the fifth author [30] showed that the isomorphism problem for virtually-free groups can be solved in doubly-exponential space when the groups are specified by context-free grammars, and in PSPACE when the groups are given by virtually-free presentations.

In the present article, we prove that the complexity bounds for the isomorphism problem in virtually-free groups can be improved significantly when one restricts attention to the class of plain groups.

▶ **Theorem 1** (Isomorphism of plain groups). *The isomorphism problem for plain groups presented by inverse-closed finite convergent length-reducing rewriting systems is in* Σ_3^{P}.

We recall that the complexity class Σ_3^{P} lies in the *polynomial hierarchy*, see for example [1, Chapter 5]:

$$\mathsf{NP} = \Sigma_1^{\mathsf{P}} \subseteq \Sigma_2^{\mathsf{P}} \subseteq \Sigma_3^{\mathsf{P}} \cdots \subseteq \mathsf{PSPACE}.$$

Here we use the following specific definition.

▶ **Definition 2** ([32]). *Let $\mathcal{S}$ be a finite set and $L \subseteq \mathcal{S}^*$. Then $L \in \Sigma_3^{\mathsf{P}}$ if and only if there is a polynomial p and a predicate P that can be evaluated in* PTIME *such that*

$$\forall w \in \mathcal{S}^* \left(w \in L \iff \exists x \in \{0,1\}^{p(|w|)}\, \forall y \in \{0,1\}^{p(|w|)}\, \exists z \in \{0,1\}^{p(|w|)} P(w,x,y,z) = 1 \right).$$

Rather than specifying the variables as polynomial length binary strings, we will describe data for x, y, z which has polynomial size over a finite alphabet.

▶ Remark 3. We will abbreviate *inverse-closed finite convergent length-reducing rewriting system* to *icfclrrs* for the rest of this article, and refer to a group admitting a presentation by an icfclrrs as an *icfclrrs group.*

▶ Remark 4. In [11] it is shown that the problem of deciding if an icfclrrs presents a plain group is in NP. Note that if the conjecture that inverse-closed finite convergent rewriting systems can only present plain groups is proved, the word "plain" may be omitted from the statement of Theorem 1.

Note that given an icfclrrs for a group, one can compute a context-free grammar for the word problem in polynomial time using the method from [8]. Hence, the results from [30] imply a doubly-exponential-space algorithm for our situation. Therefore, Theorem 1 represents a significant improvement for this special case. In [11] the second and third authors gave a bound of PSPACE for isomorphism of plain groups given as icfclrrss. This PSPACE algorithm builds upon new geometric and algebraic characterisations of icfclrrs groups developed in the same paper. Theorem 1 is again a significant improvement on this, lowering the complexity to the third level of the polynomial hierarchy. The proof of Theorem 1 combines the new characterisations of icfclrrs groups from [11], which enable us to understand maximal finite subgroup structure and conjugacy of finite order elements in these groups, with work of Babai and Szemerédi [3] to test isomorphism of finite groups efficiently using straight-line programs.

Let us briefly give a high-level intuition of the proof of Theorem 1. Verifying the ranks of the free factors of each group are the same is straightforward, so for this brief description let us assume the two plain groups are simply free products of n finite groups. We existentially guess generating sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and $\mathcal{B}_1, \ldots, \mathcal{B}_n$ for the finite factors in each group such that $|\mathcal{A}_i| = |\mathcal{B}_i|$ and mapping each $\mathcal{A}_i$ to $\mathcal{B}_i$ defines an isomorphism (in other words, we guess an isomorphism defined on generating sets), then, using straight-line programs (and methods from [3]), we universally verify that our guess indeed defines an isomorphism (that $\phi(g)\phi(h) = \phi(gh)$ for all g, h). Technically this is an infinite universal branching – still, the results of [11] ensure that considering polynomial-length straight-line programs suffice to verify that we guessed an isomorphism correctly.

However, be aware that we also need to verify that the sets we guessed, indeed, generate each group – and this is actually the more difficult part. In order to do so, we check that every element of finite order (universal branching) is conjugate to an element in the subgroup generated by some $\mathcal{A}_i$ (existential branching). Again by results in [11] we can restrict to polynomial-length straight-line programs in both the universal and existential branching. Moreover, testing whether an element has finite order can be done in polynomial time. This leads to a Σ_3^{P} algorithm.

Outline. The article is organised as follows. In Section 2, we provide background information on rewriting systems, and state the key algebraic results from [11] we need for the present work. In Section 3, we review the necessary background on straight-line programs for groups. In Section 4, we formulate a result which allows us to verify when two finite subgroups of two (potentially infinite) groups are related by an isomorphism, based on a result of Babai and Szemerédi [3]. Section 5 is devoted to a proof of our main result, Theorem 1.

Notation. Throughout this article we write log to mean $\log_2$. For $n \in \mathbb{N}_+$ we write $[1, n]$ for the interval $\{1, \ldots, n\} \subseteq \mathbb{N}$. If $\mathcal{S}$ is an *alphabet* (a non-empty finite set), we write $\mathcal{S}^*$ for the set of finite-length words over $\mathcal{S}$, and $|u|$ for the length of the word $u \in \mathcal{S}^*$; the empty word, λ, is the unique word of length 0. For a group G, we write e_G for the identity element of G.

2 Finite convergent length-reducing rewriting systems

2.1 Rewriting systems and groups

Let $\mathcal{S}$ be a generating set for a group G. If $v, w \in (\mathcal{S} \cup \mathcal{S}^{-1})^*$ and $g, h \in G$, then we write $v =_G g$ if the product of letters in v equals g; we write $v = w$ if v and w are identical as words, and $g = h$ if g and h represent the same element of G. If $v =_G g$ we say that v *spells* g. For example, the identity element e_G is spelled by the empty word λ, by aa^{-1} for any $a \in \mathcal{S}$, and so on. For an integer $r \geqslant 0$, we define the *ball of radius r in G with respect to the generating set $\mathcal{S}$*, denoted as $B_{e_G}(r)$, to be the set of all elements $g \in G$ for which there exists a word in $(\mathcal{S} \cup \mathcal{S}^{-1})^*$ of length at most r that spells g. For example, if G is the free abelian group $\langle a, b \mid ab = ba\rangle$ generated by $\mathcal{S} = \{a, b, a^{-1}, b^{-1}\}$ then the ball of radius 2 is the set of thirteen elements

$$\{e_G, a, b, a^{-1}, b^{-1}, a^2, ab, b^2, a^{-1}b, a^{-2}, a^{-1}b^{-1}, b^{-2}, ab^{-1}\}.$$

We briefly recall some basic facts concerning finite convergent length-reducing rewriting systems necessary for our discussion. We refer the reader to [4] for a broader introduction. A length-reducing rewriting system is a pair $(\mathcal{S}, T)$, where $\mathcal{S}$ is a non-empty alphabet, and T is a subset of $\mathcal{S}^* \times \mathcal{S}^*$, called a set of *rewriting rules*, such that for all $(\ell, r) \in T$ we have that $|\ell| > |r|$. We write $\mathscr{r}_T = \max_{(\ell,r)\in T}\{|r|\}$.

The set of rewriting rules determines a relation $\to$ on the set $\mathcal{S}^*$ as follows: $a \to b$ if $a = u\ell v$, $b = urv$, and $(\ell, r) \in T$. The reflexive and transitive closure of $\to$ is denoted $\xrightarrow{*}$. A word $u \in \mathcal{S}^*$ is *irreducible* if no factor is the left-hand side of any rewriting rule, and hence $u \xrightarrow{*} v$ implies that $u = v$.

The reflexive, transitive and symmetric closure of $\to$ is an equivalence denoted $\overset{*}{\leftrightarrow}$. The operation of concatenation of representatives is well defined on the set of $\overset{*}{\leftrightarrow}$-classes, and hence makes a monoid $M = M(\mathcal{S}, T)$. We say that M is the *monoid presented by* $(\mathcal{S}, T)$. When the equivalence class of every letter (and hence also the equivalence class of every word) has an inverse, the monoid M is a group and we say it is *the group presented by* $(\mathcal{S}, T)$. We note that if a rewriting system $(\mathcal{S}, T)$ presents a group G, then $\langle \mathcal{S} \mid \ell = r \text{ for } (\ell, r) \in T\rangle$ is a group presentation for G. We say that $(\mathcal{S}, T)$ (or just $\mathcal{S}$) is *inverse-closed* if for every $a \in \mathcal{S}$, there exists $b \in \mathcal{S}$ such that $ab \xrightarrow{*} \lambda$. Clearly, M is a group when $\mathcal{S}$ is inverse-closed.

A rewriting system $(\mathcal{S}, T)$ is *finite* if $\mathcal{S}$ and T are finite sets, and *terminating* (or *noetherian)* if there are no infinite sequences of allowable factor replacements. It is clear that length-reducing rewriting systems are terminating. A rewriting system is *confluent* if whenever $w \xrightarrow{*} x$ and $w \xrightarrow{*} y$, there exists $z \in \mathcal{S}^*$ such that x and y both reduce to z. A rewriting system is called *convergent* if it is terminating and confluent. In some literature, finite convergent length-reducing rewriting systems are called finite *Church-Rosser Thue* systems.

We define the *size* of a rewriting system $(\mathcal{S}, T)$ to be $n_T = |\mathcal{S}| + \sum_{(\ell,r)\in T} |\ell r|$, and we note that $r_T \leqslant n_T$.

2.2 Plain groups represented as rewriting systems

If $G_1, \ldots, G_n$ are groups with each G_i presented by $\langle \mathcal{S}_i \mid R_i \rangle$ for pairwise disjoint $\mathcal{S}_1, \ldots, \mathcal{S}_n$, the *free product* $G_1 * \cdots * G_n$ is the group presented by $\langle \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n \mid R_1 \cup \cdots \cup R_n \rangle$. A group is *plain* if it is isomorphic to the free product

$$A_1 * A_2 * \cdots * A_p * F_r$$

where p, r are non-negative integers, each A_i is a finite group and F_r is the free group of rank r.

We first observe that every plain group admits a presentation by an icfclrrs (see for example [12, Corollary 2]).

▶ **Lemma 5.** *If G is a plain group, then G admits a presentation by a finite convergent length-reducing rewriting system $(\mathcal{S}, T)$ such that $\mathcal{S} = \mathcal{S}^{-1}$ and the left-hand side of every rule has length 2.*

The following fact follows easily from the normal form theory of free products (see for example [20]).

▶ **Lemma 6.** *Two plain groups given as*

$$A_1 * A_2 * \cdots * A_p * F_r \quad \text{and} \quad B_1 * B_2 * \cdots * B_q * F_s$$

are isomorphic if and only if $p = q, r = s$ and there is a permutation σ such that $A_i \cong B_{\sigma(i)}$ for every $i \in [1, p]$.

The following proposition collects key results about icfclrrs groups proved in [11]. Recall the definitions of r_T, n_T above.

▶ **Proposition 7** ([11, Proposition 15, Lemmas 12, 8, 18]). *If G is a plain group presented by an icfclrrs $(\mathcal{S}, T)$, then*
1. *every finite subgroup H of G is conjugate to a subgroup in $B_{e_G}(r_T + 2)$;*
2. *the number of conjugacy classes of maximal finite subgroups in G is bounded above by n_T^2;*
3. *if $g, h \in B_{e_G}(r_T + 2) \setminus \{e_G\}$ are conjugate elements of finite order and $t \in G$ is such that $tgt^{-1} = h$, then $t \in Be_G(5r_T + 4)$;*
4. $\log(|B_{e_G}(r_T + 2)|) \leqslant n_T^2$.

We also make use of the following facts about finite subgroup membership.

▶ **Lemma 8** ([11, Lemma 11]). *Let G be a plain group. For $g, h \in G$ define $g \sim h$ if gh has finite order. Then*
1. *the relation $\sim$ is transitive on the set of non-trivial finite-order elements in G;*
2. *a set $\mathcal{A} = \{a_1, \ldots, a_m\} \subseteq G \setminus \{e_G\}$ generates a finite subgroup if and only if for all $i \in [1, m]$ both a_i and $a_1 a_i$ have finite order;*
3. *if A is a finite subgroup of G and $g, h \in G$ with $g \in A \setminus \{e_G\}$, then $h \in A$ if and only if h and gh have finite order.*

2.3 Algorithms for groups in rewriting systems

Next we observe that deciding if elements have finite order can be done in polynomial time.

▶ **Lemma 9** (Narendran and Otto [24, Theorem 4.8]). *There is a deterministic polynomial-time algorithm for the following problem: given an icfclrrs $(\mathcal{S}, T)$ presenting a group G and a word $u \in \mathcal{S}^*$, decide whether or not u spells an element of finite order in G. The running time is polynomial in $|T| + |u| + \sum_{(r,\ell)\in T} |\ell|$, so polynomial in $|u| + n_T$.*

By computing the Smith normal form of a matrix associated to $(\mathcal{S}, T)$, we have an efficient way to compute the number of infinite cyclic factors of a plain group given by an icfclrrs.

▶ **Lemma 10.** *There is a deterministic polynomial-time algorithm for the following problem: given an icfclrrs $(\mathcal{S}, T)$ presenting a group G, compute the torsion-free rank of the abelian group $G/[G, G]$. The running time is polynomial in n_T, and the torsion-free rank is bounded above by n_T.*

Proof. Let G_{ab} denote $G/[G, G]$, the abelianization of G. Let r denote the *torsion-free rank* of G_{ab}, which is the number of factors $\mathbb{Z}$ in the free product decomposition of G. We may compute the torsion-free rank of the abelianization G_{ab} from $(\mathcal{S}, T)$ in time that is polynomial in n_T as follows. Let $\mathcal{S}' \subseteq \mathcal{S}$ be a subset comprising exactly one generator from each pair of inverses. The information in $(\mathcal{S}, T)$ may be recorded in the form of a group presentation $\langle \mathcal{S}' \mid R\rangle$, where R interprets each rewriting rule in T as a relation over the alphabet $\mathcal{S}' \cup (\mathcal{S}')^{-1}$. The information in the presentation $\langle \mathcal{S}' \mid R \cup \{[a, b] \mid a, b \in \mathcal{S}'\}\rangle$ for G_{ab} may be encoded in an $|R| \times |\mathcal{S}'|$ matrix of integers M. These integers record the exponent sums of generators in each relation. The Smith normal form matrix S corresponding to M may be computed in time that is polynomial in the size of the $|R| \times |\mathcal{S}'|$ matrix and its entries (see, for example, [17, 33]), so polynomial in n_T. The torsion-free rank of G_{ab} is the number of zero entries along the diagonal of S (see, for example, [25, pp. 376-377]). Note that this means $r \leqslant n_T$. ◀

3 Straight-line programs

We use *straight-line programs* (or more precisely *straight-line sequences*) to represent the elements of a group A with finite generating set $\mathcal{A} = \{a_1, \dots, a_m\}$, see [31, Section 1.2.3] or [3, Section 3] for more details; we briefly recall this concept here. Let $X = \{x_1, \dots, x_m\}$ be a set of abstract symbols of size m. A straight-line program Y of rank m and length d on X is a sequence $Y = (s_1, \dots, s_d)$ where for each $i \in [1, d]$ either $s_i \in X \cup \{\lambda\}$, or $s_i = s_j s_k$ for some $j, k < i$, or $s_i = s_j^{-1}$ for some $j < i$. One says the straight-line program Y *yields* the word $w = s_d \in (X \cup X^{-1})^*$, which we also denote by $Y(x_1, \dots, x_m) = w(x_1, \dots, x_m)$. We write $Y(a_1, \dots, a_m) \in (\mathcal{A} \cup \mathcal{A}^{-1})^*$ for the word that is constructed by replacing every occurrence of $x_i^{\pm 1}$ in $Y(x_1, \dots, x_m)$ by $a_i^{\pm 1}$. We call the element $g \in A$ such that $Y(a_1, \dots, a_m) =_G g$ the *evaluation* of Y in A (with respect to $\mathcal{A}$).

An efficient way to store the straight-line program is to write instead the operations that define the elements $s_1, \dots, s_m$ of the sequence (cf. [31, p. 10]): for example, a generator $s_i = x$ is stored as the pair $(x, +)$, $s_i = \lambda$ is stored as $(\lambda, +)$, an inverse $s_i = s_j^{-1}$ is stored as $(j, -)$, and a product $s_i = s_j s_k$ is stored as (j, k). We call this sequence of operations a *straight-line sequence*. The word $Y(a_1, \dots, a_m)$ can then be computed by following the construction described in this straight-line sequence and replacing every generator x_j by a_j. To store this sequence we simply store the address (an integer in $[1, d]$ in binary) and the instruction (an integer in $[1, m]$ or at most two integers in $[1, d]$ in binary); thus a straight-line sequence of rank m and length d requires $\mathcal{O}\left(d(\log(d)+\log(m))\right)$ bits. In what follows, a straight-line program will always be represented by a straight-line sequence, and we write Y both for a straight-line program and the straight-line sequence representing it.

▶ **Example 11.** Consider the infinite cyclic group G generated by $\mathscr{A}=\{a\}$. The straight-line sequence $Y=(y_0=(x,+), y_1=(0,0), y_2=(1,1), y_3=(2,2), y_4=(3,3), y_5=(4,2), y_6=(5,0), y_7=(6,-))$ yields the word $Y(x)=x^{-21}$ in $(X\cup X^{-1})^*$ with $X=\{x\}$, and $Y(a)$ yields the element a^{-21} of G. The straight-line sequence $Y=((\lambda,+))$ yields $Y(x)=\lambda$, so $Y(a)=_G e_G$.

Every element of a finitely generated group with finite generating set $\mathscr{A}$ can be described by a straight-line sequence: one could first list $\mathscr{A}\cup\mathscr{A}^{-1}$ using $y_{2i-1}=(x_i,+)$ and $y_{2i}=(2i-1,-)$ for $i\in[1,|\mathscr{A}|]$, then choose a word that spells the desired element, and finally construct it letter-by-letter using $y_k=(k-1,j)$ (where $j=2i-1$ if the next letter is x_i, and $j=2i$ if the next letter is x_i^{-1}). However, Example 11 demonstrates that we can sometimes be more efficient than that. In fact, the following result shows that elements of a finite group always have short straight-line sequences, with respect to any given generating set.

▶ **Lemma 12** (Babai and Szemerédi [3, Lemma 7], Babai [2]). *Let A be a finite group with generating set $\mathscr{A}=\{a_1,\dots,a_m\}$. For each $g\in A$, there exists a straight-line sequence Y of rank m and of length at most $(\log|A|+1)^2$ such that $Y(a_1,\dots,a_m)=_G g$.*

If $P=(p_1,\dots,p_c), Q=(q_1,\dots,q_d)$ are two straight-line sequences of rank m and length c,d respectively, then we use the notation $[PQ]$ to denote the straight-line sequence of rank m and length $c+d+1$ defined as

$$[PQ]=(p_1,\dots,p_c,q_1\dots,q_d,(c,c+d)).$$

We call this the *product* of P and Q, since by construction if $P(x_1,\dots,x_m)=u$ and $Q(x_1,\dots,x_m)=v$ then $[PQ](x_1,\dots,x_m)=uv$. We denote by $[PQR]$ the straight-line sequence $[[PQ]R]$ of rank m and length $c+d+e+2$ where R has rank m and length e.

3.1 Compressed word problem

We note that in the setting of groups presented by icfclrrss, we can efficiently solve the word problem when the input is a straight-line sequence representing a group element.

▶ **Lemma 13** (Compressed word problem). *There is a deterministic algorithm for the following problem: given an icfclrrs $(\mathcal{S},T)$ presenting a group G, a set $\mathscr{A}=\{a_1,\dots,a_m\}\subseteq\mathcal{S}^*$ generating a subgroup $A\leqslant G$ such that $A\subseteq B_{e_G}(K)$ for some $K\in\mathbb{N}$, a word $u\in\mathcal{S}^*$ such that $u=_G g\in G$, and a straight-line sequence Y of rank m and length d, decide whether or not $Y(a_1,\dots,a_m)=_A g$.*

The running time is polynomial in $K+n_T+|u|+d+m+\max_i|a_i|$. In particular, if K is bounded by a polynomial in the input size, the algorithm runs in polynomial time.

Proof. For each $v\in\mathcal{S}^*$, let v^{-1} denote the formal inverse of v obtained by reversing and replacing each letter $x\in\mathcal{S}$ by $x^{-1}\in\mathcal{S}$.

Assume $Y=(y_1,\dots,y_d)$ where each $y_i=(x_j,+),(j,-)$ or (j,k). For $i\in[1,d]$ we compute and store a word $s_i\in\mathcal{S}^*$ of length at most K as follows:

- if $y_i=(x_j,+)$, set $s_i=a_j$;
- if $y_i=(j,-)$ with $j<i$, set $s_i=s_j^{-1}$;
- if $y_i=(j,k)$ for $j,k<i$, set s_i to be the reduced word obtained from s_js_k by applying rewriting rules.

Finally, return *true* if $s_d u^{-1}$ reduces to λ, and *false* otherwise.

Notice that that no s_i becomes longer than K. Therefore, each s_i can be computed in time polynomial in K plus the size of the rewriting system and the other data. ◀

4 Isomorphism testing for finite subgroups

In this section we describe an argument based on Babai and Szemerédi's work [3] which we require for proving Theorem 1. For now our setting is that we are given two groups (later these will be presented by rewriting systems) which come with efficient (polynomial time) algorithms to solve the word problem. Each group will contain some specified finite subgroup, say A in the first group and B in the second. We aim to verify in polynomial time the existence of an isomorphism from A to B. We start with the following well-known facts.

▶ **Lemma 14.** *Every finite group A has a generating set of size at most* $\log |A|$.

Proof. If $\{g_1, \dots, g_m\}$ is a minimal generating set and A_n is the group generated by $\{g_1, \dots, g_n\}$ for $n \in [1, m]$, then $|A_1| \geqslant 2$ and $|A_{n+1}| \geqslant 2|A_n|$ for $n \in [1, m-1]$, so $|A| \geqslant 2^m$ by induction. ◀

▶ **Lemma 15.** *Let A and B be groups. A map $f \colon A \to B$ is a group homomorphism if and only if $f(e_A) = e_B$ and $f(g)f(h)f((gh)^{-1}) = e_B$ for all $g, h \in A$*

Proof. If f is a homomorphism, then these conditions hold. Conversely, the second condition with $g = e_A$ yields $f(h)f(h^{-1}) = e_B$, so $f(h^{-1}) = f(h)^{-1}$ and $e_B = f(g)f(h)f((gh)^{-1}) = f(g)f(h)f(gh)^{-1}$. Thus, $f(g)f(h) = f(gh)$ for all $g, h \in G$. ◀

We now state the key technical result, which is the essence of [3, Proposition 4.8] where the isomorphism problem for finite groups in the so-called *black-box model* is shown to be in Σ_3^{P}.

▶ **Proposition 16** (Isomorphism between finite subgroups). *Let A, B be finite groups and $K \in \mathbb{N}_+$. Let $\mathscr{A} = \{a_1, \dots, a_m\}, \mathscr{B} = \{b_1, \dots, b_m\}$ be generating sets for A, B respectively, with $m \leqslant K$.*

Assume that for each $g \in A$ there is a straight-line program Y_g of rank m and length at most K such that $Y_g(a_1, \dots, a_m) =_A g$, and likewise for $g \in B$ there is a straight-line program Z_g of rank m and length at most K such that $Z_g(b_1, \dots, b_m) =_B g$.

Then the map $\psi : \mathscr{A} \to \mathscr{B}$ with $a_i \mapsto b_i$ induces an isomorphism $A \to B$ if and only if

$$Y(a_1, \dots, a_m) =_A e_A \iff Y(b_1, \dots, b_m) =_B e_B \tag{1}$$

for every straight-line program Y of rank m and length at most $3K + 2$.

Proof. If ψ induces an isomorphism, clearly (1) holds for all straight-line programs.

For the converse, assume that (1) holds on all rank-m straight-line programs up to length $3K + 2$.

Without loss of generality, assume $Y_{e_A} = Z_{e_B} = ((\lambda, +))$ and

$$Y_{a_i}(x_1, \dots, x_m) = Z_{b_i}(x_1, \dots, x_m) = ((x_i, +))$$

for each $i \in [1, m]$. So $Y_{e_A}(a_1, \dots, a_m) =_A e_A$, $Z_{e_B}(b_1, \dots, b_m) =_B e_B$, $Y_{a_i}(a_1, \dots, a_m) =_A a_i$ and $Z_{b_i}(b_1, \dots, b_m) =_B b_i$ for $i \in [1, m]$.

Define a map $\phi \colon A \to B$ as follows: for $g \in A$, evaluate $Y_g(b_1, \dots, b_m)$ to get an element $h \in B$, then set $\phi(g) = h$. Thus, ϕ maps each a_i to b_i.

First, by way of contradiction suppose ϕ is not a homomorphism. Since $\phi(e_A) = e_B$ by definition, Lemma 15 shows that there must exist $g, h \in A$ such that $\phi(g)\phi(h)\phi((gh)^{-1}) \neq e_B$. This means that in A we have

$$Y_g(a_1, \dots, a_m)Y_h(a_1, \dots, a_m)Y_{(gh)^{-1}}(a_1, \dots, a_m) =_A gh(gh)^{-1} =_A e_A,$$

whereas in B we have

$$Y_g(b_1, \dots, b_n)Y_h(b_1, \dots, b_n)Y_{(gh)^{-1}}(b_1, \dots, b_n) =_B \phi(g)\phi(h)\phi((gh)^{-1}) \neq_B e_B.$$

Let $Y = [Y_g Y_h Y_{(gh)^{-1}}]$ be the straight-line program of rank m and length at most $3K + 2$. Then Y contradicts our assumption that (1) holds on all rank-m straight-line programs up to length $3K + 2$. Thus ϕ is a homomorphism.

Next we show that ϕ is injective. If $g \in \ker \phi$, then $Y_g(b_1, \dots, b_m)$ evaluates to e_B; by assumption, (1) holds on input Y_g, so $g =_A Y_g(a_1, \dots, a_m) =_A e_A$ and ϕ is injective. So we have shown that ϕ is a monomorphism which satisfies $\phi : a_i \mapsto b_i$.

Repeating the preceding argument for $\phi' \colon B \to A$ defined as: for $g \in B$, evaluate $Z_g(a_1, \dots, a_m)$ to get an element $h \in A$, then set $\phi'(g) = h$; we obtain a monomorphism ϕ' with $\phi' : b_i \mapsto a_i$. Since A, B are finite this implies that $|A| = |B|$ hence the monomorphism ϕ is an isomorphism, and since $\phi(a_i) = b_i$ for $i \in [1, m]$ we have that ϕ is the (unique) isomorphism induced by ψ. ◀

▶ Remark 17. Using Lemma 13, we can check condition (1) in Proposition 16 in polynomial time in groups presented by icfclrrss.

5 Proof of the main theorem

The algorithm for the proof of our main theorem checks the conditions of the following proposition. We remark that verifying that some collection of finite subgroups are maximal and that every finite order element is conjugate to an element in one of these maximal finite subgroups turns out to be the main bottleneck for the complexity of our algorithm. These are items (2) and (3) in the following proposition.

▶ **Proposition 18.** *Let G, H be plain groups presented by icfclrrs $(\mathcal{S}, T)$, $(\mathcal{S}', T')$ respectively. Then $G \cong H$ if and only if there are subgroups $A_i \leqslant G, B_i \leqslant H$ for $i \in [1, p]$ such that the following conditions (1)–(5) are satisfied:*

(1) *for each $i \in [1, p]$ we have $A_i \subseteq B_{e_G}(\mathscr{r}_T + 2)$ and $B_i \subseteq B_{e_H}(\mathscr{r}'_T + 2)$ (in particular, they are finite subgroups).*

(2) *each A_i (resp. B_i) is a maximal finite subgroup of G (resp. H).*

(3) *every $g \in G \setminus \{e_G\}$ (resp. $h \in H \setminus \{e_H\}$) of finite order can be conjugated into exactly one A_i (resp B_i).*

(4) *for each $i \in [1, p]$ we have $A_i \cong B_i$.*

(5) *the torsion-free rank of $G/[G, G]$ is equal to the torsion-free rank of $H/[H, H]$.*

Moreover, we may choose minimal generating sets $\mathscr{A}_i \subseteq B_{e_G}(\mathscr{r}_T + 2)$, $\mathscr{B}_i \subseteq B_{e_H}(\mathscr{r}'_T + 2)$ for A_i, B_i respectively, $i \in [1, p]$ so that for all $i \in [1, p]$:

(6) $|\mathscr{A}_i| = |\mathscr{B}_i| = m_i \leqslant \log |A_i|$.

(7) *if $\mathscr{A}_i = \{a_{i,j} \mid j \in [1, m_i]\}$, $\mathscr{B}_i = \{b_{i,j} \mid j \in [1, m_i]\}$, the map $a_{i,j} \mapsto b_{i,j}$ for $j \in [1, m_i]$ induces an isomorphism $A_i \to B_i$.*

Finally, we may replace conditions (1)–(3) by:

(8) *for every $g \in B_{e_G}(\mathscr{r}_T + 2) \setminus \{e_G\}$ (resp. $h \in B_{e_H}(\mathscr{r}'_T + 2) \setminus \{e_H\}$) and every $i \in [1, p]$, if g (resp. h) and $ga_{i,1}$ (resp. $hb_{i,1}$) have finite order, then $g \in A_i$ (resp. $h \in B_i$).*

(9) *every $g \in B_{e_G}(\mathscr{r}_T + 2) \setminus \{e_G\}$ (resp. $h \in B_{e_H}(\mathscr{r}'_T + 2) \setminus \{e_H\}$) of finite order can be conjugated into exactly one A_i (resp B_i); moreover, g can be conjugated into that A_i (resp B_i) by a conjugating element of length at most $5\mathscr{r}_T + 4$ (resp. $5\mathscr{r}'_T + 4$).*

(10) *for every $g \in B_{e_G}(\mathscr{r}_T + 2) \setminus \{e_G\}$ (resp. $h \in B_{e_H}(\mathscr{r}'_T + 2) \setminus \{e_H\}$), if g (resp. h) and $ga_{i,1}$ (resp. $hb_{i,1}$) have finite order, then $ga^{\epsilon}_{i,j} \in B_{e_G}(\mathscr{r}_T + 2)$ (resp. $hb^{\epsilon}_{i,j} \in B_{e_H}(\mathscr{r}'_T + 2)$) for every $j \in [1, m_i]$ and $\epsilon \in \{\pm 1\}$.*

Proof. First, assume that $G \cong H$. We will show that conditions (1)–(7) are satisfied. By Lemma 6 there exists an isomorphism $\psi\colon G \to H$ and free product decompositions

$$G \cong A_1 * A_2 * \cdots * A_p * F_r \quad \text{and} \quad H \cong B_1 * B_2 * \cdots * B_p * F_r$$

such that $A_i \cong B_i = \psi(A_i)$ for each $i \in [1, p]$. Moreover, by Proposition 7 (item 1) we may assume A_i, B_i each lie within the balls of radius $\mathscr{r}_T + 2, \mathscr{r}'_T + 2$ in the Cayley graphs of $(G, \mathscr{S}), (H, \mathscr{S}')$ respectively. By Lemma 14 there exist minimal generating sets $\mathscr{A}_i$ and $\mathscr{B}_i$ for A_i, B_i for $i \in [1, p]$ with $|\mathscr{A}_i| = |\mathscr{B}_i| \leqslant \log |A_i|$ (we may assume without loss of generality that we choose minimal generating sets to be of the same size). Since $\psi(A_i) = B_i$ we may without loss of generality choose generators so that condition (7) holds. The normal form theory for free products ([20]) gives that: for any $i \neq j$, $A_i \cap A_j = \{e_G\}$ (resp. $B_i \cap B_j = \{e_H\}$); if $p = 0$ then G and H are free groups, and if $p \neq 0$, there are exactly p conjugacy classes of non-trivial maximal finite subgroups in G (resp. H) and they are represented by $A_1, \ldots, A_p$ (resp. $B_1, \ldots, B_p$). Condition (3) follows immediately. Condition (5) follows immediately from the fact that $G/[G, G] \cong H/[H, H]$.

Conversely, suppose there are subgroups $A_i \leqslant G, B_i \leqslant H$ for $i \in [1, p]$ such that conditions (1)–(5) are satisfied. Conditions (2) and (3) give that every maximal finite subgroup in G (resp. H) is conjugate to exactly one of the subgroups $A_1, \ldots, A_p$ (resp. $B_1, \ldots, B_p$). Since G (resp. H) is a plain group, it follows that $G \cong A_1 * \cdots * A_p * F_r$ (resp. $H = B_1 * \cdots * B_p * F_s$) for some free group of rank r (resp. s). Condition (4) gives that $A_1 \cong B_1, \ldots, A_p \cong B_p$. Condition (5) gives that $r = s$ and $F_r \cong F_s$. Thus we have that $G \cong H$.

Now let us show that conditions (1)–(3) may be replaced by conditions (8)–(10). First suppose that conditions (1)–(7) are satisfied. Lemma 8 (item 3) implies condition (8). Condition (3) and Proposition (7) (item 3) imply condition (9). Condition (1) and Lemma 8(item 3) imply condition (10).

Now suppose that conditions (4)–(10) are satisfied. To establish that condition (10) implies condition (1), suppose A_i contains an element p which lies outside $B_{e_G}(\mathscr{r}_T + 2)$. Let $u \in \mathscr{A}^*$ be a word spelling p. Then there exists a word u_1, an element $a_{i,j} \in \mathscr{A}$ and $\epsilon \in \{\pm 1\}$ so that $u_1 a^{\epsilon}_{i,j}$ is a prefix of u such that u_1 spells an element that lies in $B_{e_G}(\mathscr{r}_T + 2)$ and $u_1 a^{\epsilon}_{i,j}$ spells an element that lies outside $B_{e_G}(\mathscr{r}_T + 2)$. It follows that $u_1 \neq_G e_G$ (since $|a_{i,j}| \leqslant \mathscr{r}_T + 2$). This contradicts condition (10). Conditions (1) and (8) together imply condition (2). Condition (9) and Proposition 7 (item 1) together imply condition (3). ◀

We are now ready to prove the main result.

Proof of Theorem 1. We describe a Σ_3^{P} algorithm which on input a pair $(\mathscr{S}, T)$, $(\mathscr{S}', T')$ of icfclrrss which are promised to present plain groups, accepts if and only if the groups are isomorphic. Let $N = \max\{\mathscr{n}_T, \mathscr{n}'_T\}$ be the input size, G the plain group presented by $(\mathscr{S}, T)$ and H the plain group presented by $(\mathscr{S}', T')$.

The algorithm needs to demonstrate the existence of some $p \in \mathbb{N}$ and subgroups $A_i \leqslant G$ and $B_i \leqslant H$ for $i \in [1,p]$ which satisfy conditions (4)–(10) of Proposition 18. We first observe the following. By Proposition 7 (item 2) there are at most $\mathpzc{n}_T^2$ (resp. $(\mathpzc{n}'_T)^2$) conjugacy classes of maximal finite subgroups in G (resp. H), so we have $p \leqslant N^2$. By Lemma 14 and Proposition 7 (item 4), if $\mathscr{A}$ (resp. $\mathscr{B}$) is a minimal generating set for a maximal finite subgroup A of G (resp. B of H), then

$$|\mathscr{A}| \leqslant \log |A| \leqslant \log(|B_{e_G}(\mathpzc{r}_T + 2)|) \leqslant \mathpzc{n}_T^2 \leqslant N^2$$

(resp. $|\mathscr{B}| \leqslant (\mathpzc{n}'_T)^2 \leqslant N^2$). By Lemma 12, for each $g \in A$ (resp. $g \in B$), there exists a straight-line sequence Y of length at most $(\log|A|+1)^2 \leqslant N^4$ (resp. $(\log|B|+1)^2 \leqslant N^4$) such that Y yields g. Moreover, if $A \cong B$, we may assume they have minimal generating sets of the same size.

We now start with the following quantified statements:

```
$\exists$ sets $\mathscr{A}_i = \{a_{i,j} \in \mathscr{S}^* \mid j \in [1,m_i], |a_{i,j}| \leqslant \mathpzc{r}_T + 2\}$,
    $\mathscr{B}_i = \{b_{i,j} \in (\mathscr{S}')^*, \mid j \in [1,m_i], |b_{i,j}| \leqslant \mathpzc{r}'_T + 2\}$
    for $i \in [1,p]$ where $p \leqslant N^2, m_i \leqslant N^2$,

    $\forall$ $(u,v) \in \mathscr{S}^* \times (\mathscr{S}')^*$, $|u| \leqslant \mathpzc{r}_T + 2, |v| \leqslant \mathpzc{r}'_T + 2$,
        $(s,s') \in \mathscr{S}^* \times (\mathscr{S}')^*$, $|s| \leqslant 5\mathpzc{r}_T + 4, |s'| \leqslant 5\mathpzc{r}'_T + 4$,
        straight-line sequences $Y_i$ of rank $m_i$ and length at most $3N^4 + 2$ for each $i \in [1,p]$,

        $\exists$ $(t,t') \in \mathscr{S}^* \times (\mathscr{S}')^*$, $|t| \leqslant 5\mathpzc{r}_T + 4, |t'| \leqslant 5\mathpzc{r}'_T + 4$,
            straight-line sequences $Z_1, Z_2$ of rank $m_i$ for some $i \in [1,p]$ and length at most $N^4$.
```

Then the following procedure (predicate) verifies conditions (4)–(10) in Proposition 18 using this data.

First, apply Lemma 10 to compute the torsion-free rank of $G/[G,G]$ and $H/[H,H]$ and verify that the rank is the same for both. This establishes condition (5) of Proposition 18.

Next, run this subroutine:

```
for $i \in [1,p]$
    for $j \in [1,m_i]$
        verify that $a_{i,j}$ and $b_{i,j}$ have finite order using Lemma 9;
    for $j \in [2,m_i]$
        verify that $(a_{i,1}a_{i,j})$ and $(b_{i,1}b_{i,j})$ have finite order using Lemma 9.
```

This verifies that $\mathscr{A}_i, \mathscr{B}_i$ generate finite subgroups by Lemma 8 (item 2). Let A_i, B_i be the names of the subgroups generated by $\mathscr{A}_i, \mathscr{B}_i$ respectively. We can assume that the algorithm guesses the $\mathscr{A}_i, \mathscr{B}_i$ to be minimal generating sets of the same size, so we can assume that condition (6) is satisfied.

Next, we show that the finite subgroups A_i, B_i actually lie inside the ball of radius $\mathpzc{r}_T + 2$ (resp. $\mathpzc{r}'_T + 2$) by verifying condition (10) of Proposition 18. Run the following subroutine.

```
if $u$ has finite order (using Lemma 9) and $u \neq_G e_G$ (reduced word for $u$ is not $\lambda$)
    for $i \in [1,p]$
        if $(ua_{i,1})$ has finite order (using Lemma 9; if so then $u \in A_i$ by Lemma 8 (item 3))
            for $j \in [1,m_i]$
                compute the reduced word $u_1$ for $ua_{i,j}$ and $u_2$ for $ua_{i,j}^{-1}$,
                verify that $|u_1|, |u_2| \leqslant \mathpzc{r}_T + 2$.
```

Repeat for the word v using the analogous procedure. This establishes condition (10).

Next, to verify condition (9) of Proposition 18, we first run this pre-step.

```
for $i \in [1,p]$
    for $k \in [1,p] \setminus \{i\}$
        verify that $(a_{i,1} s a_{k,1} s^{-1})$ and $(b_{i,1} s' b_{k,1} (s')^{-1})$ have infinite order using
        Lemma 9.
```

This shows that no conjugate of $a_{k,1}$ lies in A_i (resp. no conjugate of $b_{k,1}$ lies in B_i) for $i \neq k$ by Lemma 8 (item 3) (note that we are running over all s, s' of length at most $5\mathscr{r}_T + 4, 5\mathscr{r}'_T + 4$, so all elements in $B_{e_G}(5\mathscr{r}_T + 4), B_{e_H}(5\mathscr{r}'_T + 4)$).

Now suppose that for some $g \in G \setminus \{e_G\}$ we have $g =_G \alpha^{-1} c \alpha$ and $g =_G \beta d \beta^{-1}$ for some $\alpha, \beta, c, d \in \mathcal{S}^*$ with $c =_G g_c \in A_i, d =_G g_d \in A_k$ and $i \neq k$. Recall that as in Lemma 8, $g \sim h$ means gh has finite order. Then $c = (\alpha\beta) c (\alpha\beta)^{-1}$ so $a_{i,1} \sim (\alpha\beta) d (\alpha\beta)^{-1}$ and $(\alpha\beta) d (\alpha\beta)^{-1} \sim (\alpha\beta) a_{k,1} (\alpha\beta)^{-1}$, so by Lemma 8 (item 1) $a_{i,1} \sim (\alpha\beta) a_{k,1} (\alpha\beta)^{-1}$ which contradicts the result of the pre-step. It follows that every $g \in G \setminus \{e_G\}$ lies in a conjugate of *at most* one subgroup A_i. Thus to show condition (9) it suffices to show that every $u \in B_{e_G}(\mathscr{r}_T + 2) \setminus \{e_G\}$ lies in a conjugate of *some* A_i (and analogously for v).

We show this with the following subroutine.

```
if $u$ has finite order (using Lemma 9) and $u \neq_G e_G$ (reduced word for $u$ is not $\lambda$)
    verify that $(u t a_{i,1} t^{-1})$ has finite order for some $i \in [1,p]$
        (using Lemma 9 with a loop over all $i \in [1,p]$).
```

Repeat all of the above for the word v using the analogous procedure (using the word t'). This establishes condition (9) of Proposition 18.

Next, we verify condition (8) of Proposition 18. Run this subroutine.

```
if $u$ has finite order (using Lemma 9) and $u \neq_G e_G$ (reduced word for $u$ is not $\lambda$)
    for $i \in [1,p]$
        if $(u a_{i,1})$ has finite order (using Lemma 9)
            verify that $Z_1(a_{i,1} \ldots, a_{i,m_i}) =_G u$ using Lemma 13.
```

This shows that if $u \sim a_{i,1}$ then g can be spelled by a word in $(\mathscr{A}_i \cup \mathscr{A}_i^{-1})^*$, and so $u =_G g \in A_i$. Repeat for v using the analogous procedure (using the straight-line sequence Z_2). This establishes condition (8) of Proposition 18.

Lastly, to verify condition (7) and hence (4) of Proposition 18, we check that

$$Y_i(a_{i,1}, \ldots, a_{i,m_i}) = e_G \iff Y_i(b_{i,1}, \ldots, b_{i,m_i}) = e_H$$

holds where the Y_i are straight-line sequences of rank m_i and length at most $3N^4 + 2$ for $i \in [1,p]$ which we run through in the universal statement. This can be done in polynomial time using Lemma 13. Then by Proposition 16 (with $K = N^4$), since we are running over all straight-line sequences Y_i of length $3N^4 + 2$ and rank m_i for all $i \in [1,p]$ we establish condition (4). ◀

6 Conclusion

We have shown that the isomorphism problem for plain groups given as icfclrrss is decidable in Σ_3^{P}. To the best of our knowledge this presents the smallest complexity bound for the isomorphism problem apart from some very special cases like abelian and free groups (in polynomial time using [17, 25, 33]) and finite groups given as Cayley tables (in quasipolynomial time [13, 22] and also in NP, and nearly linear time for almost all orders [9]).

There is one obvious open question: can the complexity actually be reduced to Σ_2^{P} or even to some smaller class? Note that the obstacle to reach Σ_2^{P} is to verify condition (2) and (3) of Proposition 18 via conditions (8) and (9).

Another topic for future research is to investigate the maximal size of a finite subgroup presented by an icfclrrs. If one could show a polynomial bound on this size, rather than the exponential bound used here, this could lead to a lower complexity. However, this question is wide open – and, probably, related to the long-standing conjecture that all groups presented by icfclrrss are plain.

References

1 Sanjeev Arora and Boaz Barak. *Computational complexity*. Cambridge University Press, Cambridge, 2009. A modern approach. doi:10.1017/CBO9780511804090.

2 László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 164–174, New York, NY, USA, 1991. Association for Computing Machinery. doi:10.1145/103418.103440.

3 László Babai and Endre Szemeredi. On the complexity of matrix group problems I. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 229–240, 1984. doi:10.1109/SFCS.1984.715919.

4 Ronald V. Book and Friedrich Otto. *String-rewriting systems*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993. doi:10.1007/978-1-4613-9771-7.

5 François Dahmani and Daniel Groves. The isomorphism problem for toral relatively hyperbolic groups. *Publ. Math. Inst. Hautes Études Sci.*, 107:211–290, 2008. doi:10.1007/s10240-008-0014-3.

6 François Dahmani and Vincent Guirardel. The isomorphism problem for all hyperbolic groups. *Geom. Funct. Anal.*, 21(2):223–300, 2011. doi:10.1007/s00039-011-0120-0.

7 Max Dehn. *Papers on group theory and topology*. Springer-Verlag, New York, 1987. Translated from the German and with introductions and an appendix by John Stillwell, With an appendix by Otto Schreier. doi:10.1007/978-1-4612-4668-8.

8 Volker Diekert. Some remarks on presentations by finite Church-Rosser Thue systems. In *STACS 87 (Passau, 1987)*, volume 247 of *Lecture Notes in Comput. Sci.*, pages 272–285. Springer, Berlin, 1987. doi:10.1007/BFb0039612.

9 Heiko Dietrich and James B. Wilson. Group isomorphism is nearly-linear time for most orders, 2021. Accepted for FOCS 2021. arXiv:2011.03133.

10 Andy Eisenberg and Adam Piggott. Gilman's conjecture. *J. Algebra*, 517:167–185, 2019. doi:10.1016/j.jalgebra.2018.09.022.

11 Murray Elder and Adam Piggott. On groups presented by inverse-closed finite convergent length-reducing rewriting systems, 2021. arXiv:2106.03445.

12 Murray Elder and Adam Piggott. Rewriting systems, plain groups, and geodetic graphs. *Theoretical Computer Science*, 903:134–144, 2022. doi:10.1016/j.tcs.2021.12.022.

13 V. Felsch and J. Neubüser. On a programme for the determination of the automorphism group of a finite group. In Pergamon J. Leech, editor, *Computational Problems in Abstract Algebra (Proceedings of a Conference on Computational Problems in Algebra, Oxford, 1967)*, pages 59–60, Oxford, 1970.

14 Robert H. Gilman, Susan Hermiller, Derek F. Holt, and Sarah Rees. A characterisation of virtually free groups. *Arch. Math. (Basel)*, 89(4):289–295, 2007. doi:10.1007/s00013-007-2206-3.

15 Mikhail Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987. doi:10.1007/978-1-4613-9586-7_3.

16 Robert H. Haring-Smith. Groups and simple languages. *Trans. Amer. Math. Soc.*, 279(1):337–356, 1983. doi:10.2307/1999388.

17 Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8(4):499–507, 1979. doi:10.1137/0208040.

18 Abraham Karrass, Alfred Pietrowski, and Donald Solitar. Finite and infinite cyclic extensions of free groups. *Journal of the Australian Mathematical Society*, 16(4):458–466, 1973. doi:10.1017/S1446788700015445.

19 Sava Krstić. Actions of finite groups on graphs and related automorphisms of free groups. *J. Algebra*, 124(1):119–138, 1989. doi:10.1016/0021-8693(89)90154-3.

20 Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1977 edition. doi:10.1007/978-3-642-61896-3.

21 Klaus Madlener and Friedrich Otto. Groups presented by certain classes of finite length-reducing string-rewriting systems. In *Rewriting techniques and applications (Bordeaux, 1987)*, volume 256 of *Lecture Notes in Comput. Sci.*, pages 133–144. Springer, Berlin, 1987. doi:10.5555/30432.30444.

22 Gary L. Miller. On the $n^{\log n}$ isomorphism technique (a preliminary report). In *STOC*, pages 51–58, New York, NY, USA, 1978. ACM. doi:10.1145/800133.804331.

23 David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoret. Comput. Sci.*, 37(1):51–75, 1985. doi:10.1016/0304-3975(85)90087-8.

24 Paliath Narendran and Friedrich Otto. Elements of finite order for finite weight-reducing and confluent Thue systems. *Acta Inform.*, 25(5):573–591, 1988.

25 Morris Newman. The Smith normal form. In *Proceedings of the Fifth Conference of the International Linear Algebra Society (Atlanta, GA, 1995)*, volume 254, pages 367–381, 1997. doi:10.1016/S0024-3795(96)00163-2.

26 Eliyahu Rips and Zlil Sela. Canonical representatives and equations in hyperbolic groups. *Invent. Math.*, 120(3):489–512, 1995. doi:10.1007/BF01241140.

27 Zlil Sela. The isomorphism problem for hyperbolic groups. I. *Ann. of Math. (2)*, 141(2):217–283, 1995. doi:10.2307/2118520.

28 Géraud Sénizergues. An effective version of Stallings' theorem in the case of context-free groups. In *Automata, languages and programming (Lund, 1993)*, volume 700 of *Lecture Notes in Comput. Sci.*, pages 478–495. Springer, Berlin, 1993. doi:10.1007/3-540-56939-1_96.

29 Géraud Sénizergues. On the finite subgroups of a context-free group. In *Geometric and computational perspectives on infinite groups (Minneapolis, MN and New Brunswick, NJ, 1994)*, volume 25 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 201–212. Amer. Math. Soc., Providence, RI, 1996. doi:10.1007/s002360050045.

30 Géraud Sénizergues and Armin Weiß. The isomorphism problem for finite extensions of free groups is in PSPACE. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 139, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

31 Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003. doi:10.1017/CBO9780511546549.

32 Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):1–22 (1977), 1976. doi:10.1016/0304-3975(76)90061-X.

33 Arne Storjohann. Near optimal algorithms for computing smith normal forms of integer matrices. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*, ISSAC '96, pages 267–274, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/236869.237084.

Centralized, Parallel, and Distributed Multi-Source Shortest Paths via Hopsets and Rectangular Matrix Multiplication

Michael Elkin ✉
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Ofer Neiman ✉
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

Consider an undirected weighted graph $G = (V, E, w)$. We study the problem of computing $(1+\epsilon)$-approximate shortest paths for $S \times V$, for a subset $S \subseteq V$ of $|S| = n^r$ sources, for some $0 < r \leq 1$. We devise a significantly improved algorithm for this problem in the entire range of parameter r, in both the classical centralized and the parallel (PRAM) models of computation, and in a wide range of r in the distributed (Congested Clique) model. Specifically, our centralized algorithm for this problem requires time $\tilde{O}(|E| \cdot n^{o(1)} + n^{\omega(r)})$, where $n^{\omega(r)}$ is the time required to multiply an $n^r \times n$ matrix by an $n \times n$ one. Our PRAM algorithm has polylogarithmic time $(\log n)^{O(1/\rho)}$, and its work complexity is $\tilde{O}(|E| \cdot n^{\rho} + n^{\omega(r)})$, for any arbitrarily small constant $\rho > 0$.

In particular, for $r \leq 0.313\ldots$, our centralized algorithm computes $S \times V$ $(1+\epsilon)$-approximate shortest paths in $n^{2+o(1)}$ time. Our PRAM polylogarithmic-time algorithm has work complexity $O(|E| \cdot n^{\rho} + n^{2+o(1)})$, for any arbitrarily small constant $\rho > 0$. Previously existing solutions either require centralized time/parallel work of $O(|E| \cdot |S|)$ or provide much weaker approximation guarantees.

In the Congested Clique model, our algorithm solves the problem in polylogarithmic time for $|S| = n^r$ sources, for $r \leq 0.655$, while previous state-of-the-art algorithms did so only for $r \leq 1/2$. Moreover, it improves previous bounds for all $r > 1/2$. For unweighted graphs, the running time is improved further to $\mathrm{poly}(\log \log n)$ for $r \leq 0.655$. Previously this running time was known for $r \leq 1/2$.

2012 ACM Subject Classification Theory of computation → Shortest paths

Keywords and phrases Shortest paths, matrix multiplication, hopsets

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.27

Funding *Michael Elkin*: Funded by ISF grant 2344/19.
Ofer Neiman: Funded by ISF grant 1817/17.

Acknowledgements We are grateful to François Le Gall for explaining us certain aspects of the algorithm of [27], and to Shaked Matar for helpful discussions.

1 Introduction

We consider the problem of computing $(1+\epsilon)$-*approximate shortest paths* (henceforth, $(1+\epsilon)$-ASP) in undirected weighted graphs $G = (V, E, w)$, $|V| = n$, for an arbitrarily small $\epsilon > 0$. We study this problem in the centralized, parallel (PRAM) and distributed (Congested Clique) models of computation. Our focus is on computing $(1+\epsilon)$-ASP for $S \times V$, for a set $S \subseteq V$ of *sources*, $|S| = n^r$, for a constant parameter $0 < r \leq 1$.

This is one of the most central, fundamental and intensively studied problems in Graph Algorithms. Most of the previous research concentrated on one of the two following scenarios: the *single-source* ASP (henceforth, approximate SSSP), i.e., the case $|S| = 1$, and the *all-pairs* ASP (henceforth, APASP), i.e., the case $S = V$.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 27; pp. 27:1–27:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We next overview most relevant previous results and our contribution in the centralized model of computation, and then turn to the PRAM and distributed models.

1.1 Centralized Model

The classical algorithm of Dijkstra solves *exact* SSSP problem in time $O(|E| + n \log n)$ [22]. Thorup [36] refined this bound to $O(|E| + n \log \log n)$ when weights are integers. Employing these algorithms for the ASP problem for $S \times V$ results in running time of $O(|S|(|E| + n \log \log n))$. In the opposite end of the spectrum, Galil and Margalit [24], Alon et al. [2] and Zwick [40] showed that one can use fast matrix multiplication (henceforth, FMM) to solve $(1 + \epsilon)$-APASP in time $\tilde{O}(n^{\omega})$, where ω is the matrix multiplication exponent. (n^{ω} is the time required to multiply two $n \times n$ matrices. The currently best-known estimate on ω is $\omega < 2.372\ldots$ [38, 25, 14, 1].)

By allowing larger approximation factors, one can achieve a running time of $\tilde{O}(n^2)$ for APASP.[1] Specifically, Cohen and Zwick [12] devised an algorithm for 3-APASP with this running time, and Baswana and Kavitha [5] refined the approximation ratio to $(2, w)$. The notation $(2, w)$ means that for a vertex pair (u, v), their algorithm provides an estimate with a multiplicative error of 2, and an additive error bounded by the maximal weight of an edge on some shortest $u - v$ path in the graph.

Cohen [11], Elkin [17], and Gitlitz and the current authors [18] also showed that one can obtain a $(1 + \epsilon, \beta \cdot w)$-approximation for the ASP problem for $S \times V$ in time $O(|E| \cdot n^{\rho} + |S| \cdot n^{1+1/\kappa})$, where $\beta = \beta(\epsilon, \kappa, \rho)$ is a quite large constant (as long as $\epsilon > 0, \rho > 0, 1/\kappa > 0$ are constant), and w is as in the result of Baswana and Kavitha [5].

However, if one insists on a purely multiplicative error of at most $1 + \epsilon$, for an arbitrarily small constant $\epsilon > 0$, then for dense graphs ($|E| = \Theta(n^2)$), the best-known running time for ASP for $S \times V$ is $\tilde{O}(\min\{|S| \cdot n^2, n^{\omega}\})$ (the first term can be achieved by running Dijsktra from every source, the second term by using $(1 + \epsilon)$-APASP). In the current paper we devise an algorithm that solves the problem in $\tilde{O}(n^{\omega(r)} + |E| \cdot n^{o(1)})$ time,[2] where $\omega(r)$ is the matrix multiplication exponent of *rectangular matrix multiplication*. That is, $n^{\omega(r)}$ is the time required to multiply an $n^r \times n$ matrix by an $n \times n$ matrix. Coppersmith [13] showed that for $r \leq 0.291$, $\omega(r) \leq 2 + o(1)$, and Le Gall and Urrutia [27] improved this bound further to $r \leq 0.313$. Denote $\alpha \geq 0.313$ as the maximal value such that $\omega(\alpha) \leq 2 + o(1)$. Therefore, our algorithm solves $(1 + \epsilon)$-ASP problem for $S \times V$ in $n^{2+o(1)}$ time, as long as $|S| = O(n^{\alpha})$. Moreover, the bound on our running time grows gracefully from $n^{2+o(1)}$ to n^{ω}, as the number of sources $|S|$ increases from n^{α} to n. When $S = V$, our bound matches the bound of Zwick [40]. See Table 1.

Furthermore, Dor et al. [15] showed that any $(2 - \epsilon)$-ASP algorithm for $S \times V$ that runs in $T(n)$ time, for any positive constant $\epsilon > 0$ and any function $T(n)$, translates into an algorithm with running time $T(O(n))$ that multiplies two Boolean matrices with dimensions $|S| \times n$ and $n \times n$. Thus, the running time of our algorithm cannot be improved by more than a factor of $n^{o(1)}$ without improving the best-known algorithm for multiplying (rectangular) Boolean matrices.

In terms of edge weights, the situation with our algorithm is similar to that with the algorithm of Zwick [40]. Both algorithms apply directly to graphs with polynomially-bounded edge weights. Nevertheless, we argue that both of them can be used in conjunction with the Klein-Sairam's reduction of weights [32] to provide the same bounds for graphs with arbitrary weights.

[1] By $\tilde{O}(f(n))$ we mean $O(f(n) \cdot \log^{O(1)} f(n))$.

[2] In fact, our result holds for arbitrary $0 < \epsilon < 1$, see Theorem 5.

Table 1 Results on $(1+\epsilon)$-ASP for $S \times V$ in the centralized model for weighted graphs (previous running time is for dense graphs).

# of sources	Our running time	Previous running time
$n^{0.1}$	$n^{2+o(1)}$	$n^{2.1}$
$n^{0.2}$	$n^{2+o(1)}$	$n^{2.2}$
$n^{0.3}$	$n^{2+o(1)}$	$n^{2.3}$
$n^{0.4}$	$n^{2.011}$	$n^{2.373}$
$n^{0.5}$	$n^{2.045}$	$n^{2.373}$
$n^{0.6}$	$n^{2.094}$	$n^{2.373}$
$n^{0.7}$	$n^{2.154}$	$n^{2.373}$
$n^{0.8}$	$n^{2.222}$	$n^{2.373}$
$n^{0.9}$	$n^{2.296}$	$n^{2.373}$
n^{1}	$n^{2.373}$	$n^{2.373}$

1.2 Parallel Model

The situation in the parallel setting (PRAM) is similar to that in the centralized setting. The first parallel $(1+\epsilon)$-SSSP algorithm with polylogarithmic time (specifically, $(\log n)^{\tilde{O}((\log 1/\rho)/\rho)}$ and $O(|E| \cdot n^{\rho})$ work, for any arbitrarily small constant parameter $\rho > 0$, was devised by Cohen [11]. Her bounds were improved in the last five years by [20, 21, 33, 3, 19], culminating in polylogarithmic time and $\tilde{O}(|E|)$ work [33, 3]. All these aforementioned algorithms are randomized, except for the deterministic algorithm of Elkin and Matar [19] that requires polylogarithmic time $(\log n)^{O(\mathbf{1}/\rho)}$ and work $\tilde{O}(|E| \cdot n^{\rho})$.

On the opposite end of the spectrum, algorithms of Galil and Margalit [24], Alon et al. [2], and Zwick [40] (based on FMM) can be used in the PRAM setting. They give rise to deterministic polylogarithmic time $\tilde{O}(n^{\omega})$ work [40] for the $(1+\epsilon)$-APASP problem.

By using sparse spanners, the algorithm of Cohen [11] in conjunction with that of Baswana and Sen [4] provides polylogarithmic time and $O(|E| \cdot n^{1/\kappa} + |S| \cdot n^{1+1/\kappa})$ work for $(2+\epsilon)\kappa$-ASP for $S \times V$, where $\kappa = 1, 2, \ldots$ is a parameter. Recently, Gitlitz and the current authors [18] also showed that one can have $(1+\epsilon, \beta \cdot w)$-ASP for $S \times V$ in polylogarithmic time and $O(|E| \cdot n^{\rho} + |S| \cdot n^{1+1/\kappa})$ work, where $\beta = \beta(\epsilon, \kappa, \rho)$ is a large constant (as long as $\epsilon, \rho, 1/\kappa > 0$ are constant), and w is as above.

Nevertheless, if one insists on a purely multiplicative error of at most $1+\epsilon$, currently best-known solutions for the ASP problem for $S \times V$ that run in polylogarithmic time require work at least $\Omega(\min\{|S| \cdot |E|, n^{\omega}\})$. Our parallel algorithm for the problem with $|S| = n^r$ sources, $0 < r \leq 1$, has polylogarithmic time $(\log n)^{O(1/\rho)}$ and work $\tilde{O}(n^{\omega(r)} + |E| \cdot n^{\rho})$, for any arbitrarily small constant $\rho > 0$. Similarly to the centralized setting, this results in work $n^{2+o(1)} + \tilde{O}(|E| \cdot n^{\rho})$, for any arbitrarily small constant $\rho > 0$, as long as $|S| = O(n^{\alpha})$, $\alpha = 0.313$, and it improves Zwick's bound [40] of n^{ω} (which applies for $(1+\epsilon)$-APASP) for all values of $r < 1$. The aforementioned reduction of [15] implies that the work complexity of our algorithm cannot be improved by more than a factor of $n^{o(1)}$ without improving the best-known centralized algorithm for multiplying (rectangular) Boolean matrices.

Our algorithm uses FMM and hopsets. The ingredient that builds hopsets is randomized, but by using a new deterministic construction of hopsets from [19], one can make it deterministic, with essentially the same bounds. As a result our ultimate $(1+\epsilon)$-ASP algorithms (both centralized and parallel ones) become deterministic.

1.3 Distributed Model

In the Congested Clique model, every two vertices of a given n-vertex graph $G = (V, E)$, may communicate in each round by a message of $O(\log n)$ bits. The running time of an algorithm is measured by the number of rounds. Computing shortest paths in this model has been extensively studied in the last decade. An exact APSP algorithm was devised in [10] with running time $O(n^{1-2/\omega}) = O(n^{0.158\ldots})$ for unweighted undirected graphs (or with $1+\epsilon$ error in weighted directed graphs), and in $\tilde{O}(n^{1/3})$ time for weighted directed graphs. The latter result was improved in [26] to $n^{0.209}$ when the weights are constant.

The first algorithm with polylogarithmic time for weighted undirected graphs was devised by [6], who showed a $(1+\epsilon)$-approximate *single-source* shortest paths algorithm. In [9], among other results, a $(1+\epsilon)$-ASP algorithm with polylogarithmic time was shown for a set of $\tilde{O}(n^{1/2})$ sources. For unweighted graphs, the running time was recently improved by [16] to poly$(\log\log n)$, with a similar restriction of $O(n^{1/2})$ sources.

In the current paper we obtain an algorithm for the $(1+\epsilon)$-ASP in the Congested Clique model for weighted undirected graphs with polylogarithmic time, for a set of $|S| = O(n^{\frac{1+\alpha}{2}}) = O(n^{0.655\ldots})$ sources. For larger sets of sources, our running time gracefully increases until it reaches $\tilde{O}(n^{0.158})$ time when $S = V$ (see Table 2). Denoting $|S| = n^r$, our algorithm outperforms the state-of-the-art bound of [9] for all $0.5 < r < 1$. In the case of unweighted graphs, we provide a similar improvement over the result of [16]: our $(1+\epsilon)$-ASP algorithm has poly$(\log\log n)$ time, allowing up to $n^{0.655}$ sources.

Table 2 Results on $(1+\epsilon)$-ASP for $S \times V$ in the Congested Clique model (for any constant $\epsilon > 0$, and hiding constants and lower order terms).

# of sources	Our running time	Running time of [9]	Running Time of [10]
$n^{0.5}$	$\tilde{O}(1)$	$\tilde{O}(1)$	$n^{0.158}$
$n^{0.6}$	$\tilde{O}(1)$	$n^{0.06}$	$n^{0.158}$
$n^{0.7}$	$n^{0.006}$	$n^{0.13}$	$n^{0.158}$
$n^{0.8}$	$n^{0.04}$	$n^{0.2}$	$n^{0.158}$
$n^{0.9}$	$n^{0.1}$	$n^{0.26}$	$n^{0.158}$
n^{1}	$n^{0.158}$	$n^{1/3}$	$n^{0.158}$

1.4 Additional Results

We also devise an algorithm for the $(1+\epsilon)$-approximate k-*nearest neighbors* (henceforth, k-NN) problem in PRAM. Here k, $1 \leq k \leq n$, is a parameter. For a vertex v, let $z_1, z_2, \ldots$ be all other vertices ordered by their distance from v in non-decreasing order, with ties broken arbitrarily. A vertex u is *in the* $(1+\epsilon)$-*approximate* k-*NN of* v if it is no farther from v than $(1+\epsilon)d_G(v, z_k)$. The objective is to compute $(1+\epsilon)$-approximate shortest paths for some set $\mathcal{P}$ of pairs of vertices, that for every vertex $u \in V$ contains at least k pairs (u, v) with v being in the $(1+\epsilon)$-approximate k-NN of v. Our algorithm for this problem applies even in *directed* weighted graphs. It requires polylogarithmic time and $\tilde{O}(\min\{n^{\omega}, k^{0.702}n^{1.882} + n^{2+o(1)}\})$ work. For $k = O(n^{0.168})$, this work is $n^{2+o(1)}$, and for $k = o(n^{0.698})$, this bound is better than n^{ω}, i.e., it improves the bound for $(1+\epsilon)$-APASP problem.

From technical viewpoint, in this result we adapt a centralized algorithm of Yuster and Zwick [39] for sparse matrix multiplication to the PRAM setting. We then employ this algorithm in conjunction with the observation due to Censor-Hillel et al. [9] that the k-NN problem boils down to computing k matrix products of sparse matrices. We generalize this observation and argue that this is the case not only for the exact k-NN problem, but also for its approximate variant.

1.5 Technical Overview

As was mentioned above, our algorithms employ hopsets. A graph $H = (V, E', w')$ is a $(1+\epsilon, \beta)$-*hopset* for a graph $G = (V, E, w)$, if for every vertex pair $u, v \in V$, we have

$$d_G(u,v) \le d^{(\beta)}_{G\cup H}(u,v) \le (1+\epsilon)d_G(u,v) \ . \tag{1}$$

Here $d^{(\beta)}_{G\cup H}(u,v)$ stands for β-*bounded distance* between u and v in $G \cup H$, i.e., the length of the shortest $u - v$ path between them with at most β edges (henceforth, β-*bounded path*).

Our algorithm is related to the algorithm of [9], designed for $(1+\epsilon)$-ASP for $S \times V$ in the distributed Congested Clique (henceforth, CC) model. Their algorithm starts with computing a $(1+\epsilon, \beta)$-hopset H for the input graph G. It then adds H to G, and creates an adjacency matrix A of $G \cup H$. It then creates a matrix B of dimensions $|S| \times n$, whose entries $B_{u,v}$, for $(u,v) \in S \times V$, are defined as $w(u,v)$ if $(u,v) \in E$, and ∞ otherwise. Then the algorithm computes distance products $B \star A, (B \star A) \star A, \ldots, (B \star A^{\beta-1}) \star A = B \star A^{\beta}$. By equation (1), $B \star A^{\beta}$ is a $(1+\epsilon)$-approximation of all distances in $S \times V$.

Censor-Hillel et al. [9] developed an algorithm for efficiently multiplying *sparse* matrices in the distributed CC model. They view the matrices $B, B\star A, \ldots, B\star A^{\beta-1}$, as sparse square $n \times n$ matrices, and as a result compute $B \star A^{\beta}$ efficiently via their (tailored to the CC model) algorithm. In particular, their algorithm does not use Strassen-like fast matrix multiplication (FMM) techniques, but rather focuses on carefully partitioning all the products that need to be computed in a naive matrix product of dimensions $|S| \times n$ by $n \times n$ among n available processors.

Our first observation is that this product can be computed much faster using best available fast *rectangular* matrix multiplication (FRMM) algorithms. This observation leads to our $(1+\epsilon)$-ASP algorithms for weighted graphs that significantly improve the state-of-the-art in all the three computational models that we consider (the centralized, PRAM, and distributed CC). We also need to convert matrix distance products into ordinary algebraic matrix products. This is, however, not difficult, and was accomplished, e.g., in [40]. We employ the same methodology (of [40]). Our algorithm then employs a fast *rectangular* MM in this model due to Le Gall [26]. This leads to our improved $(1+\epsilon)$-ASP algorithms in the distributed CC model (cf. Table 2).

Remarkably, while so far hopsets were used extensively in parallel/distributed/dynamic/streaming settings [11, 7, 34, 28, 29, 20, 21, 9], there were no known applications of hopsets in the classical centralized setting. Our results demonstrate that this powerful tool is extremely useful in the classical setting as well.

1.6 Organization

After reviewing some preliminary results in Section 2, we describe our algorithm for $(1+\epsilon)$-ASP for $S \times V$ in the standard centralized model in Section 3. In Section 5 we provide our algorithm for $(1+\epsilon)$-ASP for $S \times V$ in the Congested Clique model that substantially improves the number of allowed sources while maintaining polylogarithmic time (and $\text{poly}(\log\log n)$ time, for unweighted graphs). In Section 6 we devise a PRAM algorithm for $(1+\epsilon)$-ASP for $S \times V$. In Section 7 we analyze the weight reduction of [31] in the context of our algorithm and the algorithm of [40]. Finally, in Appendix A we describe our PRAM algorithm for approximate distances to k-NN.

2 Preliminaries

Matrix Multiplication and Distance Product. Fix an integer n. For $0 \leq r \leq 1$, let $w(r)$ denote the exponent of n in the number of algebraic operations required to compute the product of an $n^r \times n$ matrix by an $n \times n$ matrix.

Let $1 \leq s, q \leq n$. Let A be an $s \times n$ matrix. We denote the entry in row i and column j of the matrix A by A_{ij}. The transpose of A is A^T. We use $*$ to denote a wildcard, e.g., the notation A_{*j} refers to the vector which is the j-th column of A. For an $n \times q$ matrix B, define the distance product $C = A \star B$ by

$$C_{ij} = \min_{1 \leq k \leq n} \{A_{ik} + B_{kj}\} \ ,$$

for $1 \leq i \leq s$ and $1 \leq j \leq q$. For a parameter $\delta > 1$, we say that C' is a δ-approximation to C if for all i, j, $C_{ij} \leq C'_{ij} \leq \delta \cdot C_{ij}$.

The following theorem is an extension of a result from [40]. The latter applies to square matrices. We extend it to rectangular matrices, and argue that it is also applicable in a parallel setting.

▶ **Theorem 1** ([40]). *Let M, R be positive integers. Let A be an $n^r \times n$ matrix and B an $n \times n$ matrix, whose entries are all in $\{1, ..., M\} \cup \{\infty\}$. Then there is an algorithm that computes a $(1 + \frac{1}{R})$-approximation to $A \star B$ in deterministic time $\tilde{O}(R \cdot n^{w(r)} \cdot \log M)$.*

Proof. It was observed in [2] that the distance product $C = A \star B$ can be computed by defining $\hat{A}_{ij} = (n+1)^{M - A_{ij}}$ and, similarly, $\hat{B}_{ij} = (n+1)^{M - B_{ij}}$. Then C can be derived from $\hat{C} = \hat{A} \cdot \hat{B}$ by $C_{ij} = 2M - \lfloor \log_{n+1} \hat{C}_{ij} \rfloor$. Since the values of entries in the matrices $\hat{A}$ and $\hat{B}$ are of size $O(M \log n)$, and each algebraic operation (when computing the standard product $\hat{A} \cdot \hat{B}$) requires $\tilde{O}(M \log n)$ time, it follows that the running time is $\tilde{O}(M \cdot n^{w(r)})$.

Next, we show that the running time can be reduced to $\tilde{O}(R \cdot \log M \cdot n^{w(r)})$, at the expense of allowing $(1 + 1/R)$-approximation of the entries of $A \star B$.

If $R \geq M$ then our algorithm computes the exact distance product in $\tilde{O}(M \cdot n^{w(r)}) = \tilde{O}(R \cdot \log M \cdot n^{w(r)})$ time, and we are done.

Thus, we henceforth assume that $M > R$. We will also assume for simplicity that both R and M are integer powers of 2. If it is not the case, we can increase them by a factor at most 2, and guarantee this property. This increases the running time by at most a constant factor.

For each integer r, $\log_2 R \leq r \leq \log_2 M$, we define scaled matrices $A'(r), B'(r)$, by setting $A'_{ij}(r) = \lceil \frac{R}{2^r} \cdot A_{ij} \rceil$, if $A_{ij} \leq 2^r$, and setting it to ∞ otherwise. The entries $B'_{ij}(r)$ are defined analogously (with respect to B). Note that $A'_{ij}(r), B'_{ij}(r) \in \{0, 1, \ldots, R\} \cup \{\infty\}$.

We then compute the product matrices $C'(r) = A'(r) \star B'(r)$, for all $\log_2 R \leq r \leq \log_2 M$. Finally, the matrix C' is computed as entry-wise minimum of all the matrices $\frac{2^r}{R} \cdot C'(r)$. Note that we invoke $O(\log M)$ distance products of matrices with entries in the range $\{0, 1, \ldots, R\} \cup \{\infty\}$, and thus the overall running time is $\tilde{O}(\log M \cdot R \cdot n^{w(r)})$.

Observe that the matrix C' is entry-wise greater or equal than the matrix $C = A \star B$. In fact, this is the case for each of the matrices $\frac{2^r}{R} \cdot C'(r)$, as

$$\begin{aligned} C_{ij} &= \min_{1 \leq k \leq n} \{A_{ik} + B_{kj}\} \\ &\leq \frac{2^r}{R} \cdot \min_{1 \leq k \leq n} \{\lceil \frac{R}{2^r} \cdot A_{ik} \rceil + \lceil \frac{R}{2^r} \cdot B_{kj} \rceil\} \leq \frac{2^r}{R} \cdot C'_{ij}(r) \ . \end{aligned}$$

For the inequality in the opposite direction, consider some fixed pair of indices i, j, and let k be the witness for C_{ij}, i.e., $C_{ij} = A_{ik} + B_{kj}$. Assume without loss of generality that $A_{ik} \leq B_{kj}$. (Otherwise the index s below needs to be defined with respect to A_{ik}.) Let s be the positive integer that satisfies $2^{s-1} \leq B_{kj} < 2^s$. (If $B_{kj} = M$, we will however set $s = \log_2 M$. If $B_{kj} = 0$, then it is easy to verify that $C_{ij} = C'_{ij} = 0$.)

If $s \leq \log_2 R$ then for $r = \log_2 R$ we have

$$\begin{aligned} C'_{ij}(r) &= \frac{2^r}{R} \cdot C'_{ij}(r) = \frac{2^r}{R} \cdot \min_{1 \leq t \leq n} \{A'_{it}(r) + B'_{tj}(r)\} \\ &= \min_{1 \leq t \leq n} \{\lceil \frac{R}{2^r} \cdot A_{it} \rceil + \lceil \frac{R}{2^r} \cdot B_{tj} \rceil\} = A_{ik} + B_{kj} = C_{ij} \ . \end{aligned}$$

(Note that all terms in the minimum above are greater or equal than $A_{ik} + B_{kj}$.)

Hence we assume that $\log_2 R < s \leq \log_2 M$. Consider $r = s$. We have

$$\begin{aligned} \frac{2^r}{R} \cdot C'_{ij}(r) &= \frac{2^r}{R} \cdot \min_{1 \leq t \leq n} \{\lceil \frac{R}{2^r} \cdot A_{it} \rceil + \lceil \frac{R}{2^r} \cdot B_{tj} \rceil\} \\ &\leq \frac{2^r}{R} \cdot (\lceil \frac{R}{2^r} A_{ik} \rceil + \lceil \frac{R}{2^r} \rceil B_{kj}) \\ &\leq \frac{2^r}{R} \left(\frac{R}{2^r} A_{ik} + \frac{R}{2^r} B_{kj} + 2 \right) \\ &= (A_{ik} + B_{kj}) + 2 \cdot \frac{2^r}{R} \ . \end{aligned}$$

Recall that $B_{kj} \geq 2^{r-1}$, and thus $2^r \leq 2(A_{ik} + B_{kj})$. Hence

$$\frac{2^r}{R} \cdot C'_{ij}(r) \leq (A_{ik} + B_{kj}) + 4 \cdot \frac{A_{ik} + B_{kj}}{R} = C_{ij} \cdot (1 + 4/R) \ .$$

Hence $C'_{ij} \leq \frac{2^r}{R} \cdot C'_{ij}(r) \leq C_{ij} \cdot (1 + 4/R)$. ◀

▶ Remark 2. The algorithm of Theorem 1 boils down to $O(\log M)$ standard matrix multiplications, and choosing the minimum value for each entry. Thus, we can also apply it in the Congested Clique and PRAM models of computation. In the Congested Clique model, naively, the overhead is $O(R \cdot \log M)$. (See also Section 5 for a refined bound.) In the PRAM model, naively, the time grows by a factor of $O(R \cdot \log M)$. On the other hand, by the Chinese Remainders' theorem, one can also replace each matrix product with entries bounded by n^R by R matrix products with entries bounded by $n^{O(1)}$, and compute these products in parallel. Hence, in fact, the PRAM running time grows by a factor of $O(\log M)$, while the work complexity grows by a factor of $O(R \cdot \log M)$.

Witnesses. Given an $s \times n$ matrix A and an $n \times q$ matrix B, an $s \times q$ matrix W is called a *witness* for $C = A \star B$ if for all i, j, $C_{ij} = A_{iW_{ij}} + B_{W_{ij}j}$. It was shown in [23, 40] how to compute the matrix W in almost the same time required to compute C (up to logarithmic factors). This holds also for a witness for C' which is a c-approximation for C (see [40, Section 8]), for some $c \geq 1$. The witness can assist us in recovering the actual paths, rather than just reporting distance estimates. Since computing witnesses is done by an appropriate distance product, these witnesses can also be efficiently computed in the PRAM model.

Hopsets. Recall the definition of hopsets in the beginning of Section 1.5. A randomized construction of hopsets was gives in [11], see also [34, 29, 20]. The following version was shown in [21].

▶ **Theorem 3** ([21]). *For any weighted undirected graph $G = (V, E)$ on n vertices and parameter $\kappa > 1$, there is a randomized algorithm running in time $\tilde{O}(|E| \cdot n^{1/\kappa})$, that computes a $(1 + \epsilon, \beta)$-hopset H with $\beta = \left(\frac{\kappa}{\epsilon}\right)^{O(\kappa)}$ of size $O(n^{1+1/\kappa})$ (for every $0 < \epsilon < 1$ simultaneously) .*

We note that [19] provides a *deterministic* construction of hopsets with similar properties. There are two differences, which have essentially no effect on our result. First, the hopbound in [19] is $\beta = \left(\frac{\log n}{\epsilon}\right)^{O(\kappa)}$. Second, the construction there accepts $\epsilon > 0$ as a part of its input. Nevertheless, their hopsets can be used to make our results in PRAM deterministic, with essentially the same parameters. Our centralized algorithm can also be made deterministic using a hopset construction from [29].

Eliminating Dependence on Aspect Ratio. The aspect ratio of a graph G is the ratio between the largest to smallest edge weight. A well-known reduction by [32] asserts that to compute $(1+\epsilon)$-approximate shortest paths in $G = (V, E)$ with $|V| = n$, it suffices to compute $(1+\epsilon)$-approximate shortest paths in a collection of at most $\tilde{O}(|E|)$ graphs $\{G_t\}$. The total number of (non-isolated) vertices in all these graphs is $O(n \log n)$, the total number of edges is $\tilde{O}(|E|)$, and the aspect ratio of each graph is $O(n/\epsilon)$. This reduction can be performed in parallel (PRAM EREW) within $O(\log^2 n)$ rounds and work $O(|E|)$. Thus it can also be done in the standard centralized model in $\tilde{O}(|E|)$ time. See also Section 7 and [20, Section 4] for more details. Since in our algorithms the dependence on the aspect ratio will be logarithmic, in the sequel we assume that $M = \text{poly}(n)$.

3 Multi-Source Shortest Paths

Let $G = (V, E, w)$ be a weighted undirected graph and fix a set of s sources $S \subseteq V$. We compute a $(1+\epsilon)$-approximation for all distances in $S \times V$, by executing Algorithm 1.

Algorithm 1 ASP(G, S, ϵ).

```
Let $H$ be an $(1+\epsilon, \beta)$-hopset for $G$;
Set $R = \beta/\epsilon$;
Let $A$ be the adjacency matrix of $G \cup H$;
Let $B^{(1)} = A_{S*}$;
for $t$ from 1 to $\beta - 1$ do
    Let $B'$ be a $(1 + 1/R)$-approximation to $B^{(t)} \star A$;
    Let $B^{(t+1)}$ be entry-wise minimum between $B^{(t)}$ and $B'$;
end for
return $B^{(\beta)}$;
```

The first step is to compute an $(1+\epsilon, \beta)$-hopset H, for a parameter $\kappa \geq 1$ with $\beta = \left(\frac{\kappa}{\epsilon}\right)^{O(\kappa)}$, as in Theorem 3. Let A be the adjacency matrix of $G \cup H$ and fix $R = \beta/\epsilon$. For every integer $1 \leq t \leq \beta$, let $B^{(t)}$ be an $s \times n$ matrix such that for all $i \in S$ and $j \in V$, $B^{(t)}_{ij}$ is a $(1 + \frac{1}{R})^{t-1}$-approximation to $d^{(t)}_{G \cup H}(i, j)$. Note that $B^{(1)} = A_{S*}$ is a submatrix of A containing only the rows corresponding to the sources S.

The following claim asserts that taking an approximate distance product of $B^{(t)}$ with the adjacency matrix yields $B^{(t+1)}$.

▷ Claim 4. Let $c, c' \geq 1$. Let A be the adjacency matrix of an n-vertex graph $G = (V, E)$, and let B be an $s \times n$ matrix (whose rows correspond to $S \subseteq V$) so that for all i, j, B_{ij} is a c-approximation to $d^{(t)}_G(i, j)$, for some positive integer t. Let $C = B \star A$ and C' be a c'-approximation to C. Then, for all i, j, C'_{ij} is a $c \cdot c'$-approximation to $d^{(t+1)}_G(i, j)$.

Proof. Consider a pair of vertices $i \in S$ and $j \in V$. By definition of the $\star$ operation, $C_{ij} = \min_{1 \le k \le n}\{B_{ik} + A_{kj}\}$. Let π be the shortest path in G from i to j that contains at most $t+1$ edges, and let $k \in V$ be the last vertex before j on π. Since B_{ik} is a c-approximation to $d_G^{(t)}(i,k)$ and A_{kj} is the edge weight of $\{k,j\}$, we have that $B_{ik} + A_{kj}$ is a c-approximation to $d_G^{(t+1)}(i,j)$. Hence $C_{ij} \le B_{ik} + A_{kj}$ is a c-approximation of $d_G^{(t+1)}(i,j)$ too. The assertion of the claim follows since $C_{ij} \le C'_{ij} \le c' \cdot C_{ij}$. ◁

Given $B^{(t)}$, we compute $B^{(t+1)}$ as a $(1+\frac{1}{R})$-approximation to $B^{(t)} \star A$. Using Theorem 1 this can be done within $\tilde{O}(R \cdot n^{w(r)})$ rounds. Thus, the total running time to compute $B^{(\beta)}$ is

$$\tilde{O}(\beta \cdot R \cdot n^{w(r)}) = \tilde{O}(n^{w(r)} \cdot (\kappa/\epsilon)^{O(\kappa)})$$

By Claim 4, $B^{(\beta)}$ is a $(1+\frac{1}{R})^{\beta-1} \le e^{\epsilon} = 1 + O(\epsilon)$ approximation to $d_{G \cup H}^{(\beta)}(u,v)$ for all $u \in S$ and $v \in V$. Since H is a $(1+\epsilon,\beta)$-hopset, the matrix $B^{(\beta)}$ is a $(1+O(\epsilon))$-approximation to $d_G(u,v)$, for all $u \in S$, and $v \in V$.

Reporting paths. For each approximate distance in $S \times V$ we can also report a path in G achieving this distance. To this end, we compute witnesses for each approximate distance product, and as in [40, Section 5] there is an algorithm that can report, for any $u, v \in V$, a path in $G \cup H$ of length at most $(1+\epsilon) \cdot d_{G \cup H}^{(\beta)}(u,v)$. In order to translate this to a path in G, we need to replace the hopset edges by corresponding paths in G. We use the fact that the hopsets of [21] have a path reporting property. That is, each hopset edge of weight W' has a corresponding path π of length W' in G, and every vertex on π stores its neighbors on the path. Thus, we can obtain a $u - v$ path in G in time proportional to its number of edges.

We conclude with the following theorem.

▶ **Theorem 5.** *Let $G = (V,E)$ be a weighted undirected graph, fix $S \subseteq V$ of size n^r for some $0 \le r \le 1$, and let $0 < \epsilon < 1$. Then for any $\kappa \ge 1$, there is a deterministic algorithm that computes a $(1+\epsilon)$-approximation to all distances in $S \times V$ that runs in time*

$$\tilde{O}(\max\{n^{w(r)} \cdot (\kappa/\epsilon)^{O(\kappa)}, |E| \cdot n^{1/\kappa}\}) \ .$$

Furthermore, for each pair in $S \times V$, a path achieving the approximate distance can be reported in time proportional to the number of edges in it.

One may choose κ as a slowly growing function of n, e.g. $\kappa = (\log\log n)/\log\log\log n$, so that $\kappa^{\kappa} \le \log n$ and $n^{1/\kappa} = n^{o(1)}$, and obtain running time $\tilde{O}(n^{\omega(r)} + |E| \cdot n^{o(1)})$ (for a constant $\epsilon > 0$). We stress that for all $r \le 0.313$, a result of [27] gives that $w(r) = 2 + o(1)$. So even for polynomially large set of sources S, with size up to $n^{0.313}$, our algorithm computes $(1+\epsilon)$-approximate distances $S \times V$ in time $n^{2+o(1)}$. In fact, for all $r < 1$, our bound improves the current bound for $(1+\epsilon)$-APASP [40].

Observe that if $r > 0.313$, then we can choose κ as a large enough constant, so that the running time to compute the hopset, which is $\tilde{O}(|E| \cdot n^{1/\kappa})$, is dominated by $n^{w(r)}$. Alternatively, if $|E| \le n^{2-\delta}$ we may choose $\kappa = 1/\delta$, so the running time to compute the hopset will be $\tilde{O}(n^2) = \tilde{O}(n^{w(r)})$ for all $0 \le r \le 1$. In both cases we obtain $\beta = (1/\epsilon)^{O(1)}$, and thus our algorithm for computing $(1+\epsilon)$-approximate shortest paths for $S \times V$ has running time $\tilde{O}(n^{w(r)}/\epsilon^{O(1)})$.

4 Approximate Distance Preservers

A direct application of our s-ASP algorithm is the problem of approximate D-preservers. Exact D-preservers were introduced in [8]. Given an unweighted n-vertex graph $G = (V, E)$ and a parameter D, a subgraph $G' = (V, H)$ of G ($H \subseteq E$) is called a *D-preserver* of G if for every pair $u, v \in V$ with $d_G(u, v) \geq D$, we have $d_{G'}(u, v) = d_G(u, v)$. It was shown in [8] that every unweighted graph (both undirected and directed) admits a D-preserver with $O(n^2/D)$ edges, and that this bound is tight. We will next describe an efficient construction of an *approximate* D-preserver that applies only for *undirected* graphs.

It is also well-known (see [37, 8]) that one can compute a D-preserver of size $O(\frac{n^2}{D}\log n)$ by sampling $O(\frac{n}{D}\log n)$ vertices S independently at random, computing a BFS tree rooted at each of them, and inserting all these trees into the ultimate D-preserver. The running time of this procedure is $\tilde{O}(\frac{m \cdot n}{D})$, where $m = |E|$.

For a pair of parameters D and $\epsilon > 0$, we say that a subgraph $G' = (V, H)$ is a $(1+\epsilon)$-*approximate D-preserver* of G if for every pair of vertices $u, v \in V$ with $d_G(u, v) \geq D$, we have $d_{G'}(u, v) \leq (1+\epsilon) \cdot d_G(u, v)$. Using our $(1+\epsilon)$-approximate s-ASP algorithm one can compute a $(1+\epsilon)$-approximate D-preserver within time $\tilde{O}(n^{w(r)})$, with $r = \frac{\log n - \log D}{\log n}$. This expression is strictly better than $\tilde{O}(n^3/D)$, for all values of D, i.e., at least for dense graphs ($m = \Theta(n^2)$), the new algorithm is always faster than the existing one.

The algorithm itself uses our $(1+\epsilon)$-ASP algorithm to compute $(1+\epsilon)$-approximate BFS trees rooted at all vertices of the sampled set S, and returns the union of them as a $(1+\epsilon)$-approximate D-preserver. For the stretch analysis, consider a pair $u, v \in V$ of vertices with $d_G(u, v) \geq D$. Let π be a shortest path between them. Since it contains at least $D+1$ vertices, with high probability at least one of the sampled vertices $s \in S$ belongs to the path. Thus the preserver G' satisfies $d_{G'}(u, s) \leq (1+\epsilon) \cdot d_G(u, s)$ and $d_{G'}(s, v) \leq (1+\epsilon) \cdot d_G(s, v)$. Thus

$$d_{G'}(u, v) \leq (1+\epsilon) \cdot (d_G(u, s) + d_G(s, v)) = (1+\epsilon) \cdot d_G(u, v) \ .$$

5 Improved ASP for $S \times V$ in the Congested Clique Model

In this section we show how to improve the $(1+\epsilon)$-ASP for $S \times V$ results of [9] and [16] in the Congested Clique model. Specifically, we show that given a weighted graph $G = (V, E)$ and a set of $S \subseteq V$ sources of size $|S| = n^r$, there is a poly$(\log n)$ time algorithm to compute $(1+\epsilon)$-ASP for $S \times V$ as long as $r < (1+\alpha)/2 \approx 0.655$. For unweighted graphs, we obtain an improved running time of poly$(\log\log n)$. More generally, for S of arbitrary size, $|S| = n^r$, the running time is given by $\tilde{O}(n^{f(r)})$, where the function $f(r)$ grows from 0 to $1 - \frac{2}{\omega} \approx 0.158$. (See Table 2 for more details.)

A polylogarithmic running time (respectively, poly$(\log\log n)$ time for unweighted graphs), was obtained only for $r \leq 1/2$ in [9] (resp., [16]). More generally, their running time for arbitrary S is $\tilde{O}(\frac{|S|^{2/3}}{n^{1/3}})$.

To achieve these improvements, we use the method of [10] combined with fast rectangular matrix multiplication in the Congested Clique model. The following theorem is from [26].

▶ **Theorem 6** ([26]). *Let $G = (V, E)$ be an n-vertex graph, and fix $0 < r \leq 1$. Let A and B be $n^r \times n$ and $n \times n$ matrices. Then there is a deterministic algorithm in the Congested Clique that computes $A \cdot B$ in $O(n^{1-2/\omega(r')})$ rounds, where r' is the solution to the equation:*

$$r' = 1 - (1-r) \cdot \omega(r') \ . \qquad (2)$$

(Recall that $\omega(r')$ is the exponent for $n^{r'} \times n$ MM.)

Using this theorem in conjunction with the reduction of Theorem 1, we obtain an approximate distance product in the Congested Clique model:

▶ **Corollary 7.** *Let $G = (V, E)$ be an n-vertex graph, and fix $0 < r \leq 1$. Let A and B be $n^r \times n$ and $n \times n$ matrices with entries in $\{1, 2, ..., M\} \cup \{\infty\}$, and fix any $R \geq 1$. Then there is a deterministic algorithm in the Congested Clique that computes a $(1+1/R)$-approximation to $A \star B$ in $O(R \cdot n^{1-2/\omega(r')} \cdot \log M)$ rounds, with r' as in* (2)*.*

In fact, Le Gall [26] showed that k pairs of $n^r \times n$ and $n \times n$ matrices can be multiplied in $O(k^{2/\omega(r')} \cdot n^{1-2/\omega(r')})$ time. As a result, we improve the estimate in Corollary 7 to $O((R \cdot \log M)^{2/\omega(r')} \cdot n^{1-2/\omega(r')})$. Indeed, as we argued in the proof of Theorem 1, such an approximate distance product can be computed by calculating $O(\log M)$ distance products of matrices with entries in $\{0, 1, \ldots, R\} \cup \{\infty\}$. Each such distance product can, in turn, be computed via $O(R)$ distance products of matrices with small entries (via Chinese Remainders' Theorem; see the discussion that follows Lemma 2.2 in [40]). Hence overall, our algorithm needs to compute distance products of $O(R \cdot \log M)$ pairs of matrices, and this requires [26] $O((R \log M)^{2/w(r')} \cdot n^{1-2/w(r')})$ time.

5.1 ASP for $S \times V$ in Weighted Graphs

Here we apply the improved rectangular MM to ASP for $S \times V$, using the method of [9]. For completeness we sketch it below. The following theorem was shown in [9], based on a construction from [21]. It provides a fast construction of a hopset with logarithmic hopbound for the Congested Clique model.

▶ **Theorem 8** ([9])**.** *Let $0 < \epsilon < 1$. For any n-vertex weighted undirected graph $G = (V, E)$, there is a deterministic construction of an $(1 + \epsilon, \beta)$-hopset H with $\tilde{O}(n^{3/2})$ edges and $\beta = O(\log n/\epsilon)$, that requires $O(\log^2 n/\epsilon)$ rounds in the Congested Clique model.*

Now, we approximately compute β-bounded distances in the graph $G \cup H$, by letting B be the adjacency matrix of $G \cup H$, and $A^{(1)}$ the $|S| \times n$ matrix of sources. (Specifically, for every pair $(u, v) \in S \times V$, the entry $A^{(1)}_{u,v}$ contains $\omega((u, v))$ if $(u, v) \in E$, and ∞ otherwise.) Define $A^{(t+1)} = A^{(t)} \star B$, and by the definition of hopset, $A^{(\beta)}_{ij}$ is a $(1 + \epsilon)$-approximation to $d_G(i, j)$ for any $i \in S$ and $j \in V$. Each product is $(1 + 1/R)$-approximately computed by Corollary 7 within $\tilde{O}(R \cdot n^{1-2/\omega(r')} \cdot \log M)$ rounds. We obtain a $(1+\epsilon)(1+1/R)^{\beta}$-approximation. We set $R = O(\frac{\log n}{\epsilon^2})$. Recall also that $\beta = O(\log n/\epsilon)$. As a result we derive the following theorem:

▶ **Theorem 9.** *Given any n-vertex weighted undirected graph $G = (V, E)$ with polynomial weights, parameters $0 < r < 1$, $0 < \epsilon < 1$, and a set $S \subseteq V$ of n^r sources, let r' be the solution to equation* (2)*. Then there is a deterministic algorithm in the Congested Clique that computes $(1 + \epsilon)$-ASP for $S \times V$ within $\tilde{O}(n^{1-2/\omega(r')}/\epsilon^{O(1)})$ rounds.*

In particular, for a constant $\epsilon > 0$, when $r < (1 + \alpha)/2 \approx 0.655$ the running time is $\tilde{O}(1)$. For $r = 0.7$, the solution is slightly smaller than $r' = 0.4$, for which $\omega(r') \approx 2.01$, and the number of rounds is $O(n^{0.006})$. When $r = 0.8$, the solution is roughly $r' = 0.59$, for which $\omega(r') \approx 2.085$, and the number of rounds is $O(n^{0.04})$. We show a few more values in the following Table 2. (Note that at $r = 1$ we converge to the result of [10] for APASP.)

5.2 ASP for $S \times V$ in Unweighted Graphs

In this section we show an improved algorithm for unweighted graphs, based on [16]. The first step of [16] was developing a fast algorithm for a sparse emulator: we say that $H = (V, F)$ is an (α, β)-*emulator* for a graph $G = (V, E)$ if for all $u, v \in V$, $d_G(u, v) \leq d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta$.

▶ **Theorem 10** ([16], Theorem 24)**.** *For any n-vertex unweighted graph $G = (V, E)$ and $0 < \epsilon < 1$, there is a randomized algorithm in the Congested Clique model that computes $(1 + \epsilon, \beta)$-emulator H with $O(n \log\log n)$ edges within $O(\log^2 \beta/\epsilon)$ rounds w.h.p., where $\beta = O(\log\log n/\epsilon)^{\log\log n}$.*

Since the emulator is so sparse, all vertices can learn all of its edges within $O(\log\log n)$ rounds. Thus every pair of distance larger than β/ϵ already has an $1 + O(\epsilon)$ approximation, just by computing all distances in H locally. It remains to handle distances at most β/ϵ.

The next tool is a bounded-distance hopset that "takes care" of small distances. We say that $H' = (V, E')$ is a $(1+\epsilon, \beta', t)$-hopset if for every pair $u, v \in V$ with $d_G(u, v) \leq t$ we have the guarantee of inequality (1).

▶ **Theorem 11** ([16], Theorem 12)**.** *There is a randomized construction of a $(1+\epsilon, \beta', t)$-hopset H' with $O(n^{3/2} \log n)$ edges and $\beta' = O(\log t/\epsilon)$ that requires $O(\log^2 t/\epsilon)$ rounds w.h.p. in the Congested Clique model.*

We use a $(1 + \epsilon, \beta', t)$-hopset H' for G with $t = \beta/\epsilon = O(\log\log n/\epsilon)^{\log\log n}$, so that $\beta' = \text{poly}(\log\log n/\epsilon)$. As before we let B be the adjacency matrix of $G \cup H'$, and $A^{(1)}$ be the $|S| \times n$ matrix of sources. Define $A^{(s+1)} = A^{(s)} \star B$. By the definition of bounded-distance hopset, $A^{(\beta')}_{ij}$ is a $(1 + \epsilon)$-approximation to $d_G(i, j)$ for any $i \in S$ and $j \in V$ with $d_G(i, j) \leq t$. Each distance product is $(1+1/R)$-approximated using the algorithm from Corollary 7 within $\tilde{O}(R \cdot n^{1-2/\omega(r')} \cdot \log M)$ rounds. We note that since G is unweighted, the maximal entry in B and in any $A^{(s)}$ is $t \cdot (1 + \epsilon)$ (one can simply ignore entries of larger weight, i.e., replace them by ∞, since they will not be useful for approximating distances at most t). So we have $\log M = \text{poly}(\log\log n)$. In the current setting we assume $r \leq \frac{1+\alpha}{2} \approx 0.655$, and so $\omega(r') = 2$. The overall approximation factor is $(1+\epsilon)(1+1/R)^{\beta'}$. We set $R = \beta'/\epsilon = \text{poly}((\log\log n)/\epsilon)$, and get overall stretch $1 + O(\epsilon)$.

We conclude with the following theorem.

▶ **Theorem 12.** *Given any n-vertex unweighted undirected graph $G = (V, E)$, any $0 < \epsilon < 1$, and a set $S \subseteq V$ of at most $O(n^{0.655\cdots})$ sources, there is a randomized algorithm in the Congested Clique that w.h.p. computes $(1+\epsilon)$-ASP for $S \times V$ within* $\text{poly}(\log\log n/\epsilon)$ *rounds.*

Dory and Parter [16] provide also a deterministic counterparts of Theorems 10 and 11. Specifically, Theorem 5 of [16] provides a deterministic algorithm for building emulators with properties listed in Theorem 10 in time $O(\frac{\log^2 \beta}{\epsilon} + (\log\log n)^4)$. Theorem 12(2) in [16] provides a deterministic algorithm for building $(1 + \epsilon, \beta', t)$-hopsets with properties listed in Theorem 11, in time $O(\frac{\log^2 t}{\epsilon} + (\log\log n)^3)$. For our choice of parameters (given above), both these expressions are $\text{poly}(\log\log n, 1/\epsilon)$. As a result, we derive a deterministic counterpart of Theorem 12, i.e., $(1 + \epsilon)$-ASP for $S \times V$ in deterministic $\text{poly}(\log\log n, 1/\epsilon)$ time, for $|S| \leq n^{0.655\ldots}$.

6 PRAM Approximate Multi-Source Shortest Paths

The algorithm of Section 3 can be translated to the PRAM model. In this model, multiple processors are connected to a single memory block, and the operations are performed in parallel by these processors in synchronous rounds. The *running time* is measured by the number of rounds, and the *work* by the number of processors multiplied by the number of rounds.

To adapt our algorithm to this model, we need to show that approximate distance products can be computed efficiently in PRAM. The second ingredient is a parallel algorithm for hopsets. For the latter, the following theorem was shown in [21]. A deterministic analogue of it was recently shown in [19].

▶ **Theorem 13** ([21]). *For any weighted undirected graph $G = (V, E)$ on n vertices and parameters $\kappa \geq 1$ and $0 < \epsilon < 1$, there is a randomized algorithm that runs in parallel time $\left(\frac{\log n}{\epsilon}\right)^{O(\kappa)}$ and work $\tilde{O}(|E| \cdot n^{1/\kappa})$, that computes a $(1+\epsilon, \beta)$-hopset with $O(n^{1+1/\kappa} \cdot \log^* n)$ edges where $\beta = \left(\frac{\kappa}{\epsilon}\right)^{O(\kappa)}$.*

Matrix multiplication in PRAM. Essentially all the known fast matrix multiplication algorithms are based on Strassen's approach of divide and conquer, and thus are amenable to parallelization [30]. In particular, these algorithms which classically require time $T(n)$, can be executed in the PRAM (EREW) model within $O(\log^2 n)$ rounds and $\tilde{O}(T(n))$ work.

As was mentioned after Theorem 1, we can apply the reduction from MM to distance product in the PRAM model. Thus, we can compute a $(1+\frac{1}{R})$-approximate distance products of an $n^r \times n$ matrix by an $n \times n$ matrix in $O(R \cdot \text{poly}(\log n))$ rounds and $\tilde{O}(R \cdot n^{w(r)})$ work.

The path-reporting mechanism can be adapted to PRAM, by running the algorithm from [40] sequentially. Since we have only β iterations, the parallel time will be only $O(\beta)$ (which is a constant independent of n, as long as κ is constant). Once we got the path in $G \cup H$, we can expand all the hopset edges in parallel. We thus have the following result.

▶ **Theorem 14.** *Let $G = (V, E)$ be a weighted undirected graph, fix $S \subseteq V$ of size n^r for some $0 \leq r \leq 1$, and let $0 < \epsilon < 1$. Then for any $\kappa \geq 1$, there is a randomized parallel algorithm that computes a $(1+\epsilon)$-approximation to all distances in $S \times V$, that runs in $\left(\frac{\log n}{\epsilon}\right)^{O(\kappa)}$ parallel time, using work*

$$\tilde{O}(\min\{n^{w(r)} \cdot (\kappa/\epsilon)^{O(\kappa)}, |E| \cdot n^{1/\kappa}\}) \ .$$

Furthermore, for each pair in $S \times V$, a path achieving the approximate distance can be reported within parallel time $(\kappa/\epsilon)^{O(\kappa)}$, and work proportional to the number of edges in it.

Note that we can set κ to be an arbitrarily large constant, and obtain a polylogarithmic time and work $\tilde{O}(n^{\omega(r)} + |E|n^{1/\kappa})$.

7 Weight Reduction

In this section we argue that our s-ASP algorithm can be used in conjunction with Klein-Sairam weight reduction [31] (see also [11, 20, 19]) to replace the factor $\log M$ in the running time of its centralized version and in the work complexity of its parallel version by a factor of $O(\log^2 n/\epsilon)$ (independent of the aspect ratio of the graph).

The weight reduction produces $\lambda = \lceil \log M \rceil$ graphs $G^{(i)} = (V^{(i)}, E^{(i)})$, $i = 1, 2, \ldots, \lambda$, each with aspect ratio at most $\lceil n/\epsilon \rceil$. The vertex set $V^{(i)}$ of $G^{(i)}$ is the set of connected components of the subgraph of G in which all edges of weight at most $\epsilon \cdot 2^i/n$ are contracted. The edge set $E^{(i)}$ contains edges between nodes of $V^{(i)}$ with weight at most 2^i. Actually, we keep in the node set $V^{(i)}$ only "active" nodes, i.e., nodes that are not isolated in $G^{(i)}$.

The vertex sets $\{V^{(i)}\}_{i=1}^{\lambda}$ form a laminar family, which can be represented by a forest $\mathcal{F}$. There is an edge in $\mathcal{F}$ between a node $C^{(i+1)} \in V^{(i+1)}$ and a node $C^{(i)} \in V^{(i)}$ if and only if $C^{(i)}$ is merged into $C^{(i+1)}$ on scale $i+1$, i.e., $C^{(i+1)}$ is a union of one or more distinct sets from $V^{(i)}$, one of which is $C^{(i)}$. (It is possible that $C^{(i+1)} = C^{(i)}$.)

Denote the exponent of the running time of our centralized s-ASP algorithm by $2 \leq \zeta \leq \omega$, i.e., the running time is $\tilde{O}(n^{\zeta}) \cdot \text{poly}(1/\epsilon) \cdot \log M$. Then, once it is invoked on all the graphs $\{G_i\}_{i=1}^{\lambda}$ created by the weight reduction, the running time becomes $\log n/\epsilon \cdot \text{poly}(1/\epsilon) \cdot \sum_{i=1}^{\lambda} \tilde{O}(n_i^{\zeta})$. We next argue that

$$\sum_{i=1}^{\lambda} n_i^{\zeta} = \tilde{O}(n^{\zeta}) \ .$$

For the forest $\mathcal{F}$ as above, we denote by $f(\mathcal{F}) = \sum_{i=1}^{\lambda} n_i^{\zeta}$ the sum over all levels of $\mathcal{F}$, where each level i contributes its number n_i of nodes (that appear on level i) in the power ζ.

Observe that if a node C is active on level i (of $\mathcal{F}$), then by a level $i+\ell$, $\ell = \lceil \log n/\epsilon \rceil$, it necessarily merges into some supernode $\hat{C} \in V^{(i+\ell)}$, $C \subset \hat{C}$ ($C \neq \hat{C}$). (This is because any edge $e \in R^{(i)}$ incident on C will necessarily be contracted on or before level $i+\ell$.)

We say that a path π in $\mathcal{F}$ between some ancestor node $C^{(j)} \in V^{(j)}$ and some descendent node $C^{(i)} \in V^{(i)}$, $i \leq j$, is a *one-child* path if each of the nodes $C^{(i+1)}, C^{(i+2)}, \ldots, C^{(j)}$ has one single child in $\mathcal{F}$ (and $C^{(i)} = C^{(i+1)} = \ldots = C^{(j)}$). Such a path is said to be *maximal* if $C^{(j)}$ is a either a root or its parent has more than one child, and $C^{(i)}$ is either a leaf or has more than one child. Note that a maximal one-child path π may also be empty, if $C^{(i+1)} \in V^{(i+1)}$ is a parent of $C^{(i)} \in V^{(i)}$, $C^{(i+1)} \neq C^{(i)}$, and $C^{(i)}$ is either a leaf or has more than one child, and $C^{(i+1)}$ has more than one child. In this case we write $\pi = (C^{(i)})$.

Now consider a forest $\hat{\mathcal{F}}$ in which each maximal one-child path $\pi = (C^{(i)}, C^{(i+1)} = C^{(i)}, \ldots, C^{(j)})$ is replaced by a one-child path of length precisely ℓ. Let $\hat{\lambda} \leq \lambda \cdot \ell$ be the number of levels in $\hat{\mathcal{F}}$. Note that

$$f(\mathcal{F}) = \sum_{i=1}^{\lambda} n_i^{\zeta} \leq f(\hat{\mathcal{F}}) = \sum_{i=1}^{\hat{\lambda}} \hat{n}_i^{\zeta} \ ,$$

where $\hat{n}_i$ is the number of nodes on level i of the forest $\hat{\mathcal{F}}$. (This is because, by induction on i, we have $n_i \leq \hat{n}_i$, for every i.)

Let $\mathcal{F}'$ be the forest $\hat{\mathcal{F}}$ in which every maximal one-child path $\pi = (C^{(i)}, \ldots, C^{(j)} = C^{(i)})$ is replaced by an empty one-child path $(C^{(i)})$. Note that in $\mathcal{F}'$, every internal node has degree at least 2. Let n'_i denote the number of nodes on level i of $\mathcal{F}'$.

Observe that as the number of leaves is n, the overall number of distinct nodes in $\mathcal{F}'$ is at most $2n-1$. Each node C of $\mathcal{F}'$ contributes at most $n^{\zeta-1}$ to the sum $f(\mathcal{F}')$. (This is because if C belongs to a level on which the number of nodes is t, the total contribution of this level is t^{ζ}. Hence each node C on this level can be charged for at most $t^{\zeta-1} \leq n^{\zeta-1}$.) Hence $f(\mathcal{F}') \leq (2n-1) \cdot n^{\zeta-1} = O(n^{\zeta})$. (Alternatively, this can be seen by noting that the maximum of the sum $f(\mathcal{F}') = \sum_i {n'}_i^{\zeta}$ subject to $\sum_i n'_i = 2n-1$ is $O(n^{\zeta})$.)

Observe that every level of $\mathcal{F}'$ is duplicated ℓ times in $\hat{\mathcal{F}}$. Hence

$$f(\hat{\mathcal{F}}) = \ell \cdot f(\mathcal{F}') = O(\log(n/\epsilon) \cdot n^{\zeta}) \ .$$

Finally, as $f(\mathcal{F}) \leq f(\hat{\mathcal{F}})$, we conclude that $f(\mathcal{F}) = O(n^{\zeta} \cdot \log(n/\epsilon))$.

Hence the overall running time of the centralized version of our s-ASP algorithm (and the work complexity of its parallel version) is

$$\log(n/\epsilon) \cdot \mathrm{poly}(1/\epsilon) \cdot \tilde{O}(f(\mathcal{F})) = \tilde{O}(n^{\zeta}) \cdot \log^2(n/\epsilon) \cdot \mathrm{poly}(1/\epsilon) \ .$$

The time complexity of its parallel version is polylogarithmic in the aspect ratio (bounded by n/ϵ) of each of the graphs G_i.

In the distributed CC model one needs to compute the connected components along with their representatives in such a way that every vertex $v \in V$ represents $O(\log n)$ components (of all scales altogether). This is achieved by making sure that whenever a number of components

$C_1, C_2, \ldots, C_h$ with $|C_1| \geq |C_2| \geq \ldots \geq |C_h|$ merge into a higher-level component $\hat{C}$, a representative of one of the clusters $C_2, \ldots, C_h$ (but not C_1) becomes the representative of $\hat{C}$. See [20, 19] for more details.

Another issue arises when one needs to return output. One can compute an MST T of G, and to compute an exact distance labeling for this MST. (In the sequential setting this can be done in near-linear time.) These distance labels will provide an $(n-1)$-approximation of distances in G. Given a pair $u, v \in V$ with a distance estimate $\delta_{u,v}$, this provides us with $O(\log_{1+\epsilon} n) = O\left(\frac{\log n}{\epsilon}\right)$ scales on which one needs to look for a $(1+\epsilon)$-approximate distance estimate $\hat{d}_{u,v}$ for this pair. Indeed, note that in scales i for which $\epsilon/n \cdot 2^i > \delta_{u,v} \geq d_G(u,v)$, the whole path will be contracted. Also, in scales i for which $n \cdot 2^i < \delta_{u,v}/(n-1) \leq d_G(u,v)$, at least one edge in the $u-v$ path will have weight larger than 2^i. Hence u and v will be in different connected components of $G^{(i)}$.

We then identify components C_u, C_v, $u \in C_u$, $v \in C_v$, on each of the relevant scales i, and fetch the distance estimate between C_u and C_v in G_i. (One also needs to add to these estimates an upper bound on $Diam(C_u) + Diam(C_v)$, which is bounded by $O(\epsilon \cdot 2^i)$.) Finally, we then return the smallest among the resulting estimates. To summarize, this requires polylog(n) time per vertex pair, and $\tilde{O}(|S| \cdot n + |E|)$ time altogether.

The same approach can be used also in the PRAM and in the CC models (details are omitted). The time complexity will still be polylogarithmic in n, and the work complexity (in PRAM) is $\tilde{O}(|E| n^{\delta})$, for an arbitrarily small $\delta > 0$.

This completes the analysis of the weight reduction.

References

1 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021. doi:10.1137/1.9781611976465.32.

2 Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262, 1997. doi:10.1006/jcss.1997.1388.

3 Alexandr Andoni, Clifford Stein, and Peilin Zhong. Parallel approximate undirected shortest paths via low hop emulators. In *STOC*, 2020.

4 S. Baswana and S. Sen. A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, volume 2719 of *LNCS*, pages 384–396. Springer, 2003.

5 Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *FOCS*, pages 591–602, 2006. doi:10.1109/FOCS.2006.29.

6 Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 7:1–7:16, 2017. doi:10.4230/LIPIcs.DISC.2017.7.

7 Aaron Bernstein. Fully dynamic (2 + epsilon) approximate all-pairs shortest paths with fast query and close to linear update time. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 693–702, 2009. doi:10.1109/FOCS.2009.16.

8 Béla Bollobás, Don Coppersmith, and Michael Elkin. Sparse distance preservers and additive spanners. *SIAM J. Discret. Math.*, 19(4):1029–1055, 2005. doi:10.1137/S0895480103431046.

9 Keren Censor-Hillel, Michal Dory, Janne H. Korhonen, and Dean Leitersdorf. Fast approximate shortest paths in the congested clique. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 74–83. ACM, 2019. doi:10.1145/3293611.3331633.

10 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 143–152, 2015.

11 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000. doi:10.1145/331605.331610.

12 Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *J. Algorithms*, 38(2):335–353, 2001. doi:10.1006/jagm.2000.1117.

13 Don Coppersmith. Rectangular matrix multiplication revisited. *J. Complex.*, 13(1):42–49, 1997. doi:10.1006/jcom.1997.0438.

14 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.

15 D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29:1740–1759, 2000.

16 Michal Dory and Merav Parter. Exponentially faster shortest paths in the congested clique. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, pages 59–68, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3382734.3405711.

17 M. Elkin. Computing almost shortest paths. In *Proc. 20th ACM Symp. on Principles of Distributed Computing*, pages 53–62, 2001.

18 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths and PRAM distance oracles in weighted graphs. *CoRR*, abs/1907.11422, 2019. arXiv:1907.11422.

19 Michael Elkin and Shaked Matar. Deterministic PRAM approximate shortest paths in polylogarithmic time and slightly super-linear work. In Kunal Agrawal and Yossi Azar, editors, *SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021*, pages 198–207. ACM, 2021. doi:10.1145/3409964.3461809.

20 Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. *SIAM J. Comput.*, 48(4):1436–1480, 2019. doi:10.1137/18M1166791.

21 Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in RNC. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019.*, pages 333–341, 2019. doi:10.1145/3323165.3323177.

22 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

23 Zvi Galil and Oded Margalit. Witnesses for boolean matrix multiplication and for transitive closure. *J. Complex.*, 9(2):201–221, 1993. doi:10.1006/jcom.1993.1014.

24 Zvi Galil and Oded Margalit. All pairs shortest distances for graphs with small integer length edges. *Inf. Comput.*, 134(2):103–139, 1997. doi:10.1006/inco.1997.2620.

25 François Le Gall. Powers of tensors and fast matrix multiplication. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303. ACM, 2014. doi:10.1145/2608628.2608664.

26 François Le Gall. Further algebraic algorithms in the congested clique model and applications to graph-theoretic problems. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, volume 9888 of *Lecture Notes in Computer Science*, pages 57–70. Springer, 2016. doi:10.1007/978-3-662-53426-7_5.

27 Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1029–1046. SIAM, 2018. doi:10.1137/1.9781611975031.67.

28 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 146–155, 2014. doi:10.1109/FOCS.2014.24.

29 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 489–498, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897638.

30 Xiaohan Huang and Victor Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complex.*, 14(2):257–299, 1998. doi:10.1006/jcom.1998.0476.

31 Philip N. Klein and Sairam Subramanian. A linear-processor polylog-time algorithm for shortest paths in planar graphs. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 259–270, 1993. doi:10.1109/SFCS.1993.366861.

32 Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *J. Algorithms*, 25(2):205–220, 1997. doi:10.1006/jagm.1997.0888.

33 Jason Li. Faster parallel algorithm for approximate shortest path. In *STOC*, 2020.

34 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 565–573, 2014. doi:10.1145/2591796.2591850.

35 Yossi Shiloach and Uzi Vishkin. Finding the maximum, merging, and sorting in a parallel computation model. *J. Algorithms*, 2(1):88–102, 1981. doi:10.1016/0196-6774(81)90010-9.

36 Mikkel Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *J. Comput. Syst. Sci.*, 69(3):330–353, 2004. doi:10.1016/j.jcss.2004.04.003.

37 Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991. doi:10.1137/0220006.

38 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898. ACM, 2012. doi:10.1145/2213977.2214056.

39 Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Trans. Algorithms*, 1(1):2–13, 2005. doi:10.1145/1077464.1077466.

40 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

A Approximate Distances to k-Nearest Neighbors in PRAM

In this section, given a weighted *directed* graph $G = (V, E)$, we focus on the task of approximately computing the distances from each $v \in V$ to its k nearest neighbors. The main observation is that we work with rather sparse matrices, since for each vertex we do not need to store distances to vertices that are not among its k nearest neighbors.

In [39] fast algorithms for sparse matrix multiplication were presented. Recall that $\alpha \in [0, 1]$ is the maximal exponent so that the product of an $n \times n^\alpha$ by $n^\alpha \times n$ matrices can be computed in $n^{2+o(1)}$ time. Currently by [27], $\alpha \geq 0.313$. Let $\gamma = \frac{\omega-2}{1-\alpha}$.

▶ **Theorem 15** ([39]). *The product of two $n \times n$ matrices each with at most m nonzeros can be computed in time*

$$\min\{O(n^\omega), m^{\frac{2\gamma}{\gamma+1}} \cdot n^{\frac{2-\alpha\gamma}{\gamma+1}+o(1)} + n^{2+o(1)}\} \ .$$

We present the following adaptation to distance products in the PRAM model. In our setting, a matrix will be sparse if it contains few non-infinity values.

▶ **Lemma 16.** *For $R \geq 1$, the $(1+\frac{1}{R})$-approximate distance product of two $n \times n$ matrices each with at most m non-infinities can be computed in parallel time $O(R \log^{O(1)} n)$ and work*

$$\tilde{O}(R \cdot \min\{n^{\omega}, m^{0.702} \cdot n^{1.18} + n^{2+o(1)}\}) \ . \tag{3}$$

Proof. The $(1+\frac{1}{R})$-approximate distance product of Theorem 1 involves $O(\log M) = O(\log n)$ standard matrix multiplications. These multiplications can be done in parallel, and we need to compute entry-wise minimum of these matrices. This can also be done very efficiently in PRAM (See e.g., [35]). By the reduction described in the proof of Theorem 1, the resulting matrices will have $O(m)$ nonzeros (and entries of size $O(n^R)$). Thus the parallel time required to compute each such multiplication is $O(R \log^{O(1)} n)$. Using the currently known bounds on ω and α, we have $\gamma \approx 0.542$. Plugging this in Theorem 15, the work required is as in (3). ◀

For an $n \times n$ matrix A, denote by $\text{trun}_k(A)$ the matrix A in which every column is truncated to contain only the smallest k entries, and ∞ everywhere else. Clearly this operation can be executed in $\text{poly}(\log n)$ parallel time and $\tilde{O}(n^2)$ work. For a vertex $i \in V$, let $N_k(i)$ be the set of k nearest neighbors of i.

▷ Claim 17. Let G be a weighted directed graph. For some $t \geq 1$, and $c, c' \geq 1$, let A be an $n \times n$ matrix such that for every $1 \leq i \leq n$ and every $j \in N_k(i)$, A_{ij} is a c-approximation to $d_G^{(t)}(i,j)$, and ∞ for $j \notin N_k(i)$. Then, if B is a c'-approximation to $A^T \star A$, then for each i and $j \in N_k(i)$, we have that B_{ij} is a $(c \cdot c')$-approximation to $d_G^{(2t)}(i,j)$.

Proof. Let h be the middle vertex on the shortest path with at most $2t$ edges between i and j (so that there are at most t edges on the sub-paths from i to h and from h to j). Since $j \in N_k(i)$, the triangle inequality implies that $h \in N_k(i)$ and $j \in N_k(h)$. Thus, A_{ih} (resp., A_{hj}) is a c-approximation to $d_G^{(t)}(i,h)$ (resp., $d_G^{(t)}(h,j)$). By definition of distance product, $(A^T \star A)_{ij} \leq c \cdot d_G^{(t)}(i,h) + c \cdot d_G^{(t)}(h,j) \leq c \cdot d_G^{(2t)}(i,j)$. So B_{ij} is a $c \cdot c'$-approximation to $d_G^{(2t)}(i,j)$. (Note also that $(A^T \star A)_{ij} \geq d_G^{(t)}(i,h) + d_G^{(t)}(h,j) = d_G(i,j)$.) ◁

Our algorithm to compute approximate shortest paths to k nearest neighbors proceeds by computing $\lceil \log k \rceil$ times an approximate distance product, truncating each time to the smallest k entries in each column. See Algorithm 2. (This algorithm is based on an analogous algorithm from [9], devised there in the context of the Congested Clique model.) One difference between our algorithm and that of [9] is that on line 4 we apply a parallel version of Yuster-Zwick's sparse matrix multiplication [39], as opposed to an algorithm due to [9] for multiplying sparse matrices in the Congested Clique model. Another difference is that we are computing approximate distance products, as opposed to [9] that compute exact distance products. The latter (exact) computation applies to the Congested Clique model, and it is not clear if it can be performed in the centralized or PRAM models.

Since each matrix has $m = O(nk)$ non-infinities, and there are only $O(\log k)$ iterations, the parallel time is $R \cdot \log^{O(1)} n$ and the total work, using the bound of (3) with $m = O(nk)$, is

$$\tilde{O}(R \cdot \min\{n^{\omega}, k^{0.702} \cdot n^{1.882} + n^{2+o(1)}\}) \ .$$

The correctness of the algorithm follows from Claim 17, as the shortest path from a vertex v to a neighbor $u \in N_k(v)$ can have at most k edges. The approximation we obtain is $(1+\frac{1}{R})^{\lceil \log k \rceil} = 1 + O(\epsilon)$. We remark that the truncation might actually remove the distance from $v \in V$ to some $u \in N_k(v)$, because the computed distances are approximate, and so

Algorithm 2 `Approx` k-`NN`(G, ϵ).

```
Let $A$ be the adjacency matrix of $G$;
Let $R = \lceil (\log k)/\epsilon \rceil$;
for $i$ from 1 to $\lceil \log k \rceil$ do
    Let $A'$ be a $(1+1/R)$-approximation to $(\text{trun}_k(A))^T \star \text{trun}_k(A)$;
    Let $A$ be entry-wise minimum between $A$ and $A'$;
end for
return $\text{trun}_k(A)$;
```

u can be replaced by a farther away vertex. Denote by $N'_k(v)$ the k vertices returned by Algorithm 2 for $v \in V$. This vertex set has the property that for every vertex $u \in N_k(v)$, there is a distinct vertex $u' \in N'_k(v)$, such that $d_G(v, u') \leq (1+\epsilon) d_G(v, u)$.

Next we provide a formal argument that shows that our algorithm computes an approximate k-NN. For a vertex $u \in V$, let $z_1(u), z_2(u), \ldots, z_{n-1}(u)$ denote the sequence of vertices in the monotonically non-decreasing order of distance from u. (Henceforth ties are broken consistently by the Ids.) Let n_u denote the number of vertices reachable from u in G. If $n_u \leq k$ then k-truncation has no effect on the computation for the vertex u, and thus the computed set $N'_k(u)$ will contain all the n_u vertices reachable from u, with distance estimates approximated up to $(1+1/R)^{\lceil \log k \rceil}$. We from now on therefore focus on the case $n_u > k$.

For an index $i = 1, 2, \ldots$, we say that a vertex v is a $(1+\epsilon)$-*replacement* of $z_i(u)$ if it satisfies $d_G(u, v) \leq (1+\epsilon) d_G(u, z_i(u))$. A $(1+\epsilon)$-*approximate* k-*NN* of u is a set S of size k that satisfies the following property: Let $i \in [k]$ be the minimum index so that $z_i(u) \notin S$, if exists. Then for each $j < i$, the computed distance estimates of $z_j(u)$ are at most $(1+\epsilon)$-approximations of the actual respective distance $d_G(u, z_j(u))$, and also S contains k distinct $(1+\epsilon)$-replacements of $z_i(u)$.

Consider the following algorithm, whose pseudocode is given by Algorithm 2. Let $B_0 = B'_0 = A_G$ be the adjacency matrix of the graph G. Let $A = \text{trun}_k(A_G)$, $A_0 = A$ be the k-truncated matrix A_G. (The entries (u, v) that survive also contain distance estimates $\delta(u, v) = w(u, v)$. In other entries the estimates are set to ∞.)

Let B'_1 be a $(1+1/R)$-approximate $A_0^T \star A_0$. For every entry (x, y) we check if $B'_1(x, y) > A_0(x, y)$. If it is the case, we set $B_1(x, y) = A_0(x, y)$. Otherwise set $B_1(x, y) = B'_1(x, y)$. Set $A_1 = \text{trun}_k(B_1)$, and iterate, i.e., repeat these operations $h = \lceil \log k \rceil$ times. The matrix A_h is the output matrix.

For every $i \in [0, h]$ and every vertex u, let $\hat{S}_u(i)$ denote the set of vertices v with $B_i(u, v) \neq \infty$, and $S_u(i)$ denote the set of vertices with $A_i(u, v) \neq \infty$. Also, let $Ball_u(i)$ denote the set of vertices v such that there exists a shortest $u - v$ path with at most 2^i hops. Let $p_u(i) = |Ball_u(i)|$. Observe that since $n_u > k$, for every $i \in [0, h-1]$ we have $p_u(i) \geq 2^i$, and $p_u(h) \geq k$. We also write $Ball'_u(i) = Ball_u(i) \cap \{z_1(u), \ldots, z_k(u)\}$, and $q_u(i) = |Ball'_u(i)|$. Note that $q_u(h) = k$.

▶ **Lemma 18.** *For every vertex $u \in V$ and index $i \in [h]$,* either

1. *The set $S_u(i)$ contains all the $q_u(i)$ vertices of $Ball'_u(i)$ themselves (with estimates that are $(1+1/R)^i$-approximations of the actual respective distances from u)*
2. Or: *Let $k_i < k$ be the smallest index such that $z_{k_i}(u) \notin S_u(i)$. Then $S_u(i)$ contains all the vertices of $\{z_1(u), \ldots, z_{k_i - 1}(u)\} \cap Ball_u(i)$ (with estimates that are $(1+1/R)^i$-approximations of the actual respective distances from u), and also, $S_u(i)$ contains k distinct $(1+1/R)^i$-replacements of $z_{k_i}(u)$.*

▶ Remark. We will use the lemma with $i = h$, and deduce that for every vertex u, the set $S_u(i)$ is a $(1 + 1/R)^h$-approximate k-NN for u. (By $outdeg(u)$ we denote the out-degree of the vertex u, i.e., the number of its outgoing neighbors.)

Proof. The proof is by induction on i.

Base. For every $u \in V$, the set $S_u(0)$ contains the $\min\{k, outdeg(u)\}$ closest neighbors to u. In particular, it contains the closest vertex $z_1(u)$, and its estimate is $\delta(u, z_1(u)) = w(u, z_1(u)) = d_G(u, z_1(u))$. Note that $Ball'_u(0) = Ball_u(0) = \{z_1(u)\}$, i.e., $q_u(0) = 1$. Thus assertion 1 holds.

Step. First suppose that assertion 2 holds for u with respect to i. Let $k_i \leq k$ be the smallest index such that $z_{k_i}(u) \notin S_u(i)$. Then all vertices of $\{z_1(u), \ldots, z_{k_i-1}(u)\} \cap Ball'_u(i)$ belong to $S_u(i)$, and their distance estimates are $(1 + 1/R)^i$-approximate ones. Also, $S_u(i)$ contains k distinct $(1 + 1/R)^i$-replacements of $z_{k_i}(u)$.

There are two cases. If $k_{i+1} \geq k_i$ then, by definition, all the vertices $\{z_1(u), \ldots, z_{k_{i+1}-1}(u)\} \cap Ball'_u(i + 1)$ belong to $S_u(i + 1)$, and their distance estimates are $(1+1/R)^{i+1}$-approximate ones. Also, the remaining elements of $S_u(i+1)$ have estimates that are smaller or equal than the estimates of the respective elements in $S_u(i)$, and thus they are $(1 + 1/R)^i$-replacements of $z_{k_i}(u)$. Hence they are also $(1 + 1/R)^{i+1}$-replacements of $z_{k_{i+1}}(u)$, and we are done.

Consider now the complementary case $k_{i+1} < k_i$. Then $S_u(i + 1)$ contains all vertices of $\{z_1(u), \ldots, z_{k_{i+1}-1}(u)\} \cap Ball'_u(i+1)$ with $(1+1/R)^{i+1}$-approximate estimates. (It is easy to verify that for every $j \in [0, h]$ and $v \in S_u(j)$, the estimate of v is a $(1 + 1/R)^j$-approximate one.) It follows that vertices from $\{z_{k_{i+1}}(u), z_{k_{i+1}+1}(u), \ldots, z_{k_i-1}(u)\}$ that belong to $S_u(i)$ were pushed out from $S_u(i + 1)$. For this to happen, the set $S_u(i + 1)$ must contain k distinct vertices with a better estimate than that of $z_{k_{i+1}}(u)$, i.e., with an estimate at most $(1 + 1/R)^i \cdot d_G(u, z_{k_{i+1}}(u))$. These k distinct vertices are $(1 + 1/R)^i$-replacements, and thus $(1 + 1/R)^{i+1}$-replacements too, of $z_{k_{i+1}}(u)$, proving that assertion 2 holds for u with respect to $i + 1$ in this case too.

Hence from now we assume that assertion 1 holds for u with respect to i. Thus, $S_u(i)$ contains all the $q_u(i)$ vertices of $Ball'_u(i)$, with estimates that may possibly be by a factor at most $(1 + 1/R)^i$ off their actual distance from u. The induction hypothesis with respect to i applies also to all these vertices $z_1(u), \ldots, z_{q_u(i)}(u)$ of $Ball'_u(i)$. [3]

For each $j \in [q_u(i)]$, let $q_j = q_{z_j(u)}(i)$.

Case 1. Suppose first that all these vertices also satisfy assertion 1 of the induction hypothesis for i, i.e., for every index $j \in [q_u(i)]$, the set $S_{z_j(u)}(i)$ contains the vertices $z_1(z_j(u)), \ldots, z_{q_j}(z_j(u))$ themselves with $(1 + 1/R)^i$-approximate estimates of their distance from $z_j(u)$.

Observe that for any vertex $z \in Ball'_u(i + 1)$, either $z \in Ball'_u(i)$, or $z \in Ball'_{z_j}(i)$ and z_j lies on a shortest $u - z$ path in G, for some index $j \in [q_u(i)]$. In both these cases, the $(1 + 1/R)$-approximate distance product computed on iteration $i + 1$ of the algorithm guarantees that the set $\hat{S}_u(i + 1)$ contains z, with a distance estimate which is at most $(1 + 1/R)(1 + 1/R)^i = (1 + 1/R)^{i+1}$ off the actual distance $d_G(u, z)$. Hence $Ball'_u(i + 1) \subseteq \hat{S}_u(i + 1)$.

[3] Actually, the indices of these vertices need not necessarily be consecutive with respect to the distance from u. But to keep the notation simple, we denote them as if they were consecutive.

Recall that $S_u(i+1)$ is the k-truncation of $\hat{S}_u(i+1)$, i.e., it contains k vertices of $\hat{S}_u(i+1)$ with the smallest estimates. If it contains all these vertices z with the aforementioned $(1+1/R)^{i+1}$-approximate estimates, then assertion 1 holds for u with respect to $i+1$. So (within Case 1) we are left with the subcase that at least one of these vertices $z \in Ball'_u(i+1)$ was pushed out of this k-truncation ($S_u(i+1)$). In the latter case, let $z' = z_r(u) = z_{k_{i+1}(u)}(u)$ be such a vertex with the smallest index r. It follows that $z_r \in S_u(i+1) \setminus Ball'_u(i+1)$, but all vertices of $Ball'_u(i+1)$ with smaller index (closer to u) belong to $S_u(i+1)$. By the above argument, these vertices have $(1+1/R)^{i+1}$-approximate distance estimates.

In addition, $S_u(i+1)$ must contain k vertices x whose estimate $\delta(u,x)$ satisfies

$$\delta(u,x) \le \delta(u,z_r) \le (1+1/R)^{i+1} \cdot d_G(u,z_r) \ .$$

As $d_G(u,x) \le \delta(u,x)$ (this inequality holds for all estimates computed by our algorithm), it follows that each such vertex x is a $(1+1/R)^{i+1}$-replacement of $z_r = z_{k_{i+1}(u)}(u)$. This completes the proof for Case 1.

Case 2. In this case $S_u(i)$ contains all the $q_u(i)$ vertices of $Ball'_u(i)$ themselves (with $(1+1/R)^i$-approximate estimates), and at least one of these vertices $z_j \in Ball'_u(i)$ satisfies assertion 2 of the induction hypothesis with respect to i.

Recall that each $z_r \in Ball'_u(i+1)$ either belongs to $Ball'_u(i)$ (and then, by the assumption of this case, to $S_u(i)$), or to $Ball'_{z_j}(i)$, for some $z_j \in Ball'_u(i)$, and a shortest $u - z_r$ path traverses z_j.

If all $z_r \in Ball'_u(i+1)$ satisfy $z_r \in S_{z_j}(i)$ for some $z_j \in Ball'_u(i)$ (and a shortest $u - z_r$ path traverses z_j), then $Ball'_u(i+1) \subseteq \hat{S}_u(i+1)$. In this case the argument that we gave in Case 1 applies, and assertion of the lemma holds for u with respect to $i+1$ as well.

Otherwise, let r be the smallest index such that $z_r = z_r(u) \in Ball'_u(i+1) \cap Ball'_{z_j}(i)$, for some $z_j \in Ball'_u(i)$, and the shortest $u - z_r$ path contains z_j, and $z_r \notin S_{z_j}(i)$. (Moreover, $z_r \notin Ball'_u(i)$, and there exists no other shortest $u - z_r$ path that traverses some $z_t \in Ball'_u(i) \subseteq S_u(i)$, such that $z_r \in S_{z_t}(i)$. Indeed, in the latter case, z_r still reaches $\hat{S}_u(i+1)$, and the argument of Case 1 is applicable to it.)

For all vertices in $Ball'_u(i+1) \cap \{z_1, \ldots, z_{r-1}\}$, by the above argument, $S_u(i+1)$ contains them with $(1+1/R)^{i+1}$-approximate estimates. Also, by the induction hypothesis applied to to z_j, the set $S_{z_j}(i)$ contains k distinct $(1+1/R)^i$-replacements x of z_r that reach $\hat{S}_u(i+1)$, and they satisfy

$$\begin{aligned} \delta(u,x) &\le (1+1/R) \cdot (\delta(u,z_j) + \delta(z_j,z_r)) \\ &\le (1+1/R)((1+1/R)^i \cdot d_G(u,z_j) + (1+1/R)^i \cdot d_G(z_j,z_r)) \\ &= (1+1/R)^{i+1} \cdot d_G(u,z_r) \ . \end{aligned}$$

Hence all these vertices are $(1+1/R)^{i+1}$-replacements for z_r, and $S_u(i+1)$ contains either them, or k distinct vertices with yet smaller estimates. Thus assertion 2 holds for u with respect to $i+1$, proving the lemma. ◀

Our algorithm can also recover the paths with approximate distances for every $i \in V$ and $j \in N'_k(i)$. This is done by applying the algorithm from [40, Section 5], while executing the recursive calls in parallel.[4]

[4] Here is a brief sketch: Recall that we compute the witnesses for all the $O(\log k)$ distance products. Given a pair $i \in V$ and $j \in N'_k(i)$, if W is the witness matrix in the last iteration of the algorithm, then there are two cases: Either W_{ij} contains the middle vertex h (with at most $k/2$ hops to both i,j) on the approximate $i - j$ path. Then we can simply recurse in parallel on the pairs i,h and h,j, and then concatenate the paths. Otherwise, when $W_{ij} = 0$, we just return the edge (i,j).

▶ **Theorem 19.** *Let $G = (V, E)$ be a weighted directed n-vertex graph, and let $1 \leq k \leq n$ and $0 < \epsilon < 1$ be some parameters. Then there is a deterministic parallel algorithm that computes a $(1 + \epsilon)$-approximation to all distances between any $u \in V$ and its k nearest neighbors, that runs in parallel time $O((\log^{O(1)} n)/\epsilon)$, using work*

$$\tilde{O}(\min\{n^{\omega}, k^{0.702} \cdot n^{1.882} + n^{2+o(1)}\}/\epsilon) \ .$$

Furthermore, for each $i \in V$ and $j \in N'_k(i)$, a path achieving the approximate distance can be reported in $O(\log k)$ parallel time and work proportional to the number of edges in it.

Note that for $k \leq n^{0.168}$ this work is $n^{2+o(1)}$, and while $k \leq n^{0.698}$ the work is smaller than n^{ω}.

Cardinality Constrained Scheduling in Online Models

Leah Epstein
Department of Mathematics, University of Haifa, Israel

Alexandra Lassota
Chair of Discrete Optimization, EPFL, Lausanne, Switzerland

Asaf Levin
Faculty of Industrial Engineering and Management, Technion, Haifa, Israel

Marten Maack
Heinz Nixdorf Institute & Department of Computer Science, Paderborn University, Germany

Lars Rohwedder
School of Business and Economics, Maastricht University, The Netherlands

Abstract

Makespan minimization on parallel identical machines is a classical and intensively studied problem in scheduling, and a classic example for online algorithm analysis with Graham's famous list scheduling algorithm dating back to the 1960s. In this problem, jobs arrive over a list and upon an arrival, the algorithm needs to assign the job to a machine. The goal is to minimize the makespan, that is, the maximum machine load. In this paper, we consider the variant with an additional cardinality constraint: The algorithm may assign at most k jobs to each machine where k is part of the input. While the offline (strongly NP-hard) variant of cardinality constrained scheduling is well understood and an EPTAS exists here, no non-trivial results are known for the online variant. We fill this gap by making a comprehensive study of various different online models. First, we show that there is a constant competitive algorithm for the problem and further, present a lower bound of 2 on the competitive ratio of any online algorithm. Motivated by the lower bound, we consider a semi-online variant where upon arrival of a job of size p, we are allowed to migrate jobs of total size at most a constant times p. This constant is called the migration factor of the algorithm. Algorithms with small migration factors are a common approach to bridge the performance of online algorithms and offline algorithms. One can obtain algorithms with a constant migration factor by rounding the size of each incoming job and then applying an ordinal algorithm to the resulting rounded instance. With this in mind, we also consider the framework of ordinal algorithms and characterize the competitive ratio that can be achieved using the aforementioned approaches. More specifically, we show that in both cases, one can get a competitive ratio that is strictly lower than 2, which is the bound from the standard online setting. On the other hand, we prove that no PTAS is possible.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Cardinality Constrained Scheduling, Makespan Minimization, Online Algorithms, Lower Bounds, Pure Online, Migration, Ordinal Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.28

Related Version *Full Version*: `https://arxiv.org/abs/2201.05113` [13]

Funding L. Epstein and A. Levin were partially supported by GIF – the German-Israeli Foundation for Scientific Research and Development (grant number I-1366-407. 6/2016 on "Polynomial Migration for Online Scheduling").
Alexandra Lassota: A. Lassota was supported by the Swiss National Science Foundation within the project *Lattice algorithms and Integer Programming* (200021_185030/1).
Asaf Levin: A. Levin was also partially supported by a grant from ISF – Israeli Science Foundation (grant number 308/18).

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 28; pp. 28:1–28:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Marten Maack: Partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" under the project number 160364472 – SFB 901/3.

1 Introduction

Scheduling jobs on identical parallel machines is a well-studied problem. Such problems were in particular investigated extensively in online settings, where the algorithm has to make decisions before the whole instance is revealed. Graham's List Scheduling from the 1960's [21] is a textbook algorithm by now and an early example of an online algorithm (although the notion of competitive analysis was not formalized at that time). In this work, we study a generalization that considers an additional cardinality constraint on the number of jobs allowed on a machine.

The Cardinality Constrained Scheduling problem. We are given a set $\mathcal{J}$ of n jobs, a set $\mathcal{M}$ of m identical parallel machines and a positive integer k. Each job j has a job size p_j, which is also known as the processing time of the job. A feasible solution is a non-preemptive schedule (each job has to be assigned as a whole) satisfying the condition that for each machine i, the number of jobs assigned to i is at most k. Our goal is to minimize the makespan, that is, the maximum completion time of any job. In the context of makespan minimization, one does not need to explicitly consider the time axis and instead, a non-preemptive schedule can be defined as a partition of the job set to m machines, that is, a function $\sigma : \mathcal{J} \to \mathcal{M}$. The *load* of machine i in schedule σ is the total size of jobs assigned to i, that is, $\sum_{j \in \sigma^{-1}(i)} p_j$. The objective is to minimize the maximum load of a machine. It is easy to see that given σ, one can construct a schedule with makespan equal to the maximum load. Summarizing, the goal for the cardinality constrained scheduling problem is to find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ such that $\max_{i \in \mathcal{M}} |\sigma^{-1}(i)| \le k$ while minimizing the makespan $C_{\max}(\sigma) = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} p_j$.

The cardinality constraint arises naturally in settings where one needs to balance not only the loads of the machines, but also the number of jobs. Suppose, for example, one wants to distribute passengers to airplanes for the same trip, but different times. The passengers' luggage weight may vary and jet fuel usage is very sensitive to excess weight. Thus, the goal is to minimize the maximum loaded airplane (assuming for simplicity that this dominates the fuel cost). Extensions of the original problem to multiple dimensions are well-known and studied in both offline and online settings, see for example the vector scheduling problem in [4]. However, in contrast to this problem, the second "dimension" in our problem is a hard constraint, which makes it much more difficult to handle.

In the offline setting, the job set is given beforehand and the goal is to find a feasible solution of minimum cost. We refer to [8, 10, 9, 23, 26, 27] for previous studies of the offline setting of the problem. Since the problem is NP-hard in the strong sense, the best possible approximation result is an efficient polynomial time approximation scheme (EPTAS), that is, an algorithm that returns a feasible solution (if one exists) of cost at most $(1 + \varepsilon)$ times the optimal cost and the time complexity is upper bounded by the product of a computable function in ε and a polynomial in the (binary) encoding length of the input. Such an algorithm was given in [8]. Surprisingly, there exists no previous work on the online setting of this problem.

Computational models studied in this work. In the online setting, the input is given as a sequence of jobs. After a job is released, the algorithm learns the properties of the job (that is, the job size) and decides on the assignment of this job. This assignment decision is irrevocable and the algorithm is forced to maintain the feasibility of the solution after the assignment of each job (as long as the input has a feasible solution). Once the job assignment is decided, the adversary constructing the input sequence learns the algorithm's assignment decision and chooses the size of the next job or stops the input sequence. The *competitive ratio* of the online algorithm is a valid upper bound on the ratio between the cost of the solution returned by the algorithm and the optimal cost of an offline algorithm that sees the entire input sequence in advance (and may run in exponential time).

The model of ordinal algorithms is different. Here, the algorithm needs to decide an assignment of n jobs to m machines without seeing the sizes of the jobs. The only information that the algorithm can access is how these job sizes relate to each other, that is, the jobs are given as a list sorted non-increasingly by their sizes. If the algorithm has decided upon the assignment σ, it means that the i-th largest job in the input sequence is assigned to machine $\sigma(i)$, and this applies for all i. We say that an ordinal algorithm has *rate* α if for every input that satisfies the ordinal assumptions, the cost of the solution constructed by the assignment of the algorithm is at most α times the optimal cost for the same input.

Further, we study the model of algorithms with constant migration factor (also known as *robust algorithms*) similar to the online setting. But, unlike in online algorithms, once a job j is released, it is also allowed to modify the schedule of a subset of jobs of total size at most $\beta \cdot p_j$ where β is the migration factor. We require that β is a constant. Usually, one cannot maintain an optimal solution using a robust algorithm. Thus, we use robust approximation algorithms. We will use the terms competitive ratio and approximation ratio interchangeably, since some of our models are intermediate between online algorithms and offline algorithms. We say that a polynomial time algorithm that treats the input as a sequence is a robust α-approximation algorithm if it has a constant migration factor and in every sequence of jobs, the resulting solution has cost of at most α times the optimal cost. Similarly, a robust PTAS is a family of algorithms containing a robust $(1+\varepsilon)$-competitive algorithm for all $\varepsilon > 0$. It is called robust EPTAS (or robust FPTAS respectively) if its running time is upper bounded by some computable function of (or a polynomial in) $\frac{1}{\varepsilon}$ times a polynomial in the binary encoding length of the input.

Results and outline of the paper. We present new results for all three of the models mentioned above. An overview can be found in Table 1. In the pure online case, we first prove a lower bound of 2 on the competitive ratio of any (deterministic) algorithm. A natural idea for an online algorithm is to create a balanced schedule, i.e., a schedule in which the property is maintained that any two machines receive approximately the same number of jobs. This should limit the adversary's options to exploit the cardinality constraint. However, we show that such an approach fails by establishing a lower bound of m for the competitive ratio of algorithms maintaining the property that the number of jobs placed on any two machines differs by at most $o(\log(k))$. Another simple approach is to use variants of Greedy algorithms such as the list scheduling algorithm, which always assigns the next job to the machine with the lowest load. One would need to stop considering a machine once it has received k jobs. However, this approach is also deemed to fail, since it may create a large inbalance in the number of jobs assigned to the machines. If for example one machine has only one job and all others are full (which could happen using list scheduling), then the competitive ratio can be $k-1$ (when the next $k-1$ jobs are huge compared to the previous ones).

We utilize these insights in the design of an intricate online algorithm with constant competitive ratio, namely 120. This algorithm avoids both lower bounds by allowing a certain inbalance in the number of jobs, which is then gradually reduced as more and more jobs arrive. These results are presented in Section 2. Furthermore, we give a tight $\frac{1+\sqrt{5}}{2}$-competitive online algorithm for the special case $m = k = 2$ in the full version of the paper, see [13].

Table 1 An overview of the main results of this paper. The results stated in the parentheses have been completely removed from this extended abstract and can be found in the full version, see [13].

Computational Model	Result
Online algorithms	120-competitive algorithm, lower bound of 2 Lower bound of m for balanced algorithms (Tight $\frac{1+\sqrt{5}}{2}$-competitive algorithm for $m = k = 2$)
Ordinal algorithms	Algoritm with rate $\frac{81}{41}$
Robust algorithms	Robust $((1+\epsilon) \cdot \frac{81}{41})$-approximation with $\frac{1+\epsilon}{\epsilon}$ migration factor (Lower bound of ≈ 1.05 for constant migration, $m \geq 3$, unbounded k) (Robust FPTAS for $m = 2, k > 1/\epsilon^2$, and robust EPTAS for constant k)

Next, we consider the mentioned relaxed online settings starting with ordinal algorithms in Section 3. There is a known lower bound of $\frac{3}{2}$ regarding ordinal algorithm for the makespan minimization problem [30] which applies to the CCS problem as well. We present an ordinal algorithm with rate $\frac{81}{41}$ for CCS which is based on spreading out the m largest jobs over all machines and then filling the machines gradually with a repeating overlapping pattern. This gives an improvement over the rate 2, which can be achieved using a very simple round robin strategy.

In Section 4, we turn our attention to robust algorithms. First, we show that an ordinal algorithm with rate at most α can be turned into a robust $((1+\epsilon)\alpha)$-approximation with migration factor $\frac{1+\epsilon}{\epsilon}$. Together with our ordinal algorithm, this shows a separation between the strict online setting (having a lower bound of 2) and the setting with migration. On the other hand, we present a lower bound of roughly 1.05 for the ratio of robust algorithms for CCS. Hence, we cannot hope for a PTAS with a constant migration factor. However, the lower bound only works for cases with $m \geq 3$ and k part of the input, and we are able to present a robust EPTAS or FPTAS for the case with constant k or $m = 2, k > 1/\epsilon^2$, respectively.

Finally, we also show in the full version of the paper, see [13], that the results of this paper cannot be extended to a generalization of CCS called Class Constrained Scheduling by showing a super-constant lower bound on the competitive ratio of robust algorithms for that problem.

All the results, proofs and details that were excluded in this extended abstract can be found in the full version of the paper, see [13].

Related work. The standard problem of makespan minimization on identical machines is obtained from the CCS problem by deleting the constraint saying that the number of jobs assigned to each machine is at most k. At first glance, it seems that letting k grow to infinity in CCS would lead to similar results to the ones known for makespan minimization on identical machines (without cardinality constraints). However, we show that this is not the case and the corresponding possible competitive ratios in our problem are significantly higher than the one achievable for the problem without cardinality constraints. This is the case for the study of online algorithms as well as for robust algorithms.

The possible competitive ratio of the online algorithm for makespan minimization on identical machines is approximately 1.92 [1, 19] whereas the best lower bound for that problem is 1.88 by Rudin [31]. For small constant number of machines, it is known that there are better algorithms, for example, two machines List Scheduling (LS) [21] has a competitive ratio of $\frac{3}{2}$. We establish a lower bound of $2 - \frac{1}{k}$ on the competitive ratio for CCS that shows that the possible competitive ratios for CCS are strictly higher than the ones achievable for the problem without cardinality constraints both in the regime of a small fixed number of machines and in the general case (in both of these scenarios, we establish a lower bound of 2 when k grows unboundedly). Makespan minimization was also studied in terms of ordinal algorithms [30] and it is known that there is an ordinal algorithm for this problem on identical machines with constant rate. In particular, for large numbers of machines m, there is an algorithm of rate at most $\frac{5}{3}$ and no algorithm has rate smaller than $\frac{3}{2}$. The model of robust algorithms was introduced in [32] for makespan minimization on identical machines, where it is shown that there is a robust polynomial time approximation scheme for this problem. Namely, for every $\varepsilon > 0$, there is a $(1 + \varepsilon)$-approximation algorithm whose migration factor is upper bounded by some function of ε.

Cardinality Constrained Bin Packing (CCBP) [2, 3, 11, 28] is the variant of CCS where the maximum job size is at most 1, the makespan is forced to be at most 1 in every feasible solution, but the algorithm is allowed to buy machines. The goal is to minimize the number of machines bought by the algorithm. The best possible competitive ratio for CCBP is 2 with respect to the absolute competitive ratio as well as with respect to the asymptotic competitive ratio [2, 3, 5]. Regarding robust algorithms, it was shown in [15] that for every fixed value of k such that $k \geq 3$, there is no asymptotic approximation scheme for CCBP with constant migration factor. Observe the difference with our results for CCS where for fixed constant value of k, we establish the existence of an approximation scheme for CCS with constant migration factor.

Ordinal algorithms were studied for other scheduling problems as well, see e.g. [12, 22, 29, 34, 35], and robust algorithms were designed and analyzed for various scheduling problems and other packing problems (see e.g. [6, 7, 14, 15, 16, 17, 18, 20, 24, 25, 33]).

Notation. Throughout the paper, log refers to a logarithm with base 2. For a job subset J, we let $p(J) = \sum_{j \in J} p_j$. For a positive integer x, we let $[x] = \{1, 2, \ldots, x\}$. Without loss of generality, we assume $\mathcal{M} = [m]$. When we consider a specific algorithm (online, ordinal, or an algorithm with constant migration), we let ALG denote the cost of the solution constructed by the algorithm, and we let OPT denote the optimal offline cost for the same instance.

2 Pure online algorithms

In this section, we study the competitive ratios of online algorithm for CCS, starting with the lower bounds and then continuing with a constant competitive algorithm.

2.1 Lower bound

We show that when considering the online problem, there is no (deterministic) algorithm that has a competitive ratio smaller than 2.

▶ **Theorem 1.** *No online algorithm for CCS has a competitive ratio strictly smaller than* 2*, and for a fixed value of k, no algorithm has a competitive ratio strictly smaller than $2 - \frac{1}{k}$.*

Proof. We assume that $m \geq k$. The input consists of two phases. In the first phase, $m \times (k - 1)$ jobs of size 1 arrive. Then the adversary examines the output of the algorithm. If the algorithm has assigned exactly $k - 1$ jobs to each machine, then the input continues with one big job of size k that is the last job of the input. Otherwise the input continues with m jobs of size N where N is some very big number. In the first case, we have a ratio of $2 - 1/k$. In the second case, the competitive ratio is at least $2N/(N + k)$. ◀

We sometimes use the intuition of every machine having k slots, each of which may contain exactly one job or may be empty. Note that the ratio in the second case gets worse if a larger number of slots remain free on some machine since the total number of free slots remaining after the arrival of the first phase is m. This gives us the intuition that an algorithm should try to balance the number of jobs each machine receives. However, this possible strategy cannot guarantee a constant competitive algorithm as the next proposition shows. The proof can be found in the full version of the paper, see [13].

▶ **Proposition 2.** *Let $t \geq 1$ be an integer number that may depend on k such that $t = o(\log k)$. Let* ALG *be an algorithm that maintains the invariant that the number of jobs placed on any two machines may differ by at most t. Then the competitive ratio of* ALG *is at least m.*

Observe that obtaining an online algorithm with a competitive ratio of $\min\{m, k\}$ is trivial, as any feasible solution has a cost that is at most $\min\{m, k\}$ times the optimal cost. Thus, this is the competitive ratio of scheduling the jobs in a round-robin manner. Hence, the lower bound of Proposition 2 for this class of algorithms means that in order to establish small competitive algorithms, we need to exhibit an algorithm that does not belong to this class. In fact, in our algorithm, we have situations where there is a pair of machines with cardinalities that differ by $\Theta(\log k)$.

2.2 A competitive algorithm for CCS

Next, we present an algorithm for CCS with a constant competitive ratio. For simplicity we assume that every job's size is a power of two, that is, $p_j = 2^i$ for some (not necessarily positive) integer i. More precisely, every incoming job is rounded down accordingly. Then the resulting makespan (and the competitive ratio) will only increase by a factor of 2 when considering the correct sizes.

The general idea of the algorithm. We start by briefly discussing the main idea of the algorithm. For simplicity of presentation, assume that we know the value $p^{\infty}_{\max}$, which is the maximum size of any job by the end of the instance.

We group jobs by size. Group G_i contains the jobs j with $p_j = p^{\infty}_{\max}/2^i$ for $i = 0, \ldots, \lfloor \log k \rfloor$. Further, group G_∞ contains all smaller jobs, that is, jobs of sizes below $p^{\infty}_{\max}/2^{\lfloor \log k \rfloor + 1} \leq p^{\infty}_{\max}/k$ (recall that the jobs are rounded to powers of 2). Consider the following approach: Each group is scheduled independently using a round-robin strategy. For each group, the first job of this size is assigned to machine 1, the next one to machine 2, etc. Once every machine has one job of G_i, we continue with machine 1 again for this group. This is done for all values of i including ∞, see Figure 1 for an illustration. This method approximately balances both the loads of the machines and their cardinalities: There are still differences between machine loads due to two reasons. First, each group may have one additional job (of the group) assigned to some machines compared to the other machines (this happens when the number of jobs of the group is not divisible by m). Second, the group G_∞ may have jobs of very different sizes. All these sizes are small with respect to the maximum job size, and they are smaller by a factor of at least k (so the total size of k such jobs, which is the maximum per machine, is still at most $p^{\infty}_{\max}$). Thus, the load of every machine is at most the average load plus an additional additive error term that is at most

$$\sum_{i=0}^{\lfloor \log k \rfloor} \frac{p^{\infty}_{\max}}{2^i} + k \cdot \frac{p^{\infty}_{\max}}{2^{\log k}} \leq 3p^{\infty}_{\max}.$$

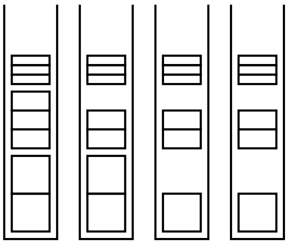


Figure 1 Example schedule of the round-robin based algorithm.

We use OPT to denote the optimal makespan for the entire input. As $p^{\infty}_{\max}$ forms a lower bound on OPT, and the average load is also a valid lower bound on OPT, the makespan is never larger than four times the optimal makespan. An issue arises once the cardinality on some machines arrives at k. In that case it is no longer feasible to schedule all groups independently. On the other hand, every pair of cardinalities (for two machines) differs by at most $\lfloor \log k \rfloor + 2$. Thus, if this difficulty occurs, then on each machine, there is only space for $O(\log k)$ more jobs (this is the number of remaining slots). If we assign the remaining jobs arbitrarily, the difference in loads can increase only by $O(\log k) \cdot p^{\infty}_{\max}$. This implies we get a $O(\log k)$-competitive algorithm assuming we know the value $p^{\infty}_{\max}$.

Indeed, the algorithm above is even 8-competitive (including the factor of 2 due to rounding) as long as no machine's cardinality is full, that is, as long as no machine has k jobs assigned to by the algorithm. On the other hand, we showed in Proposition 2 that no competitive algorithm can maintain a constant difference in the cardinalities of the job sets of any two machines, or even a difference of $o(\log k)$. Nevertheless, we manage to obtain a constant competitive algorithm by modifying the approach stated above further. Towards this we gradually decrease the number of groups G_i as the machines get filled more and more, so that by the time some machine is full (i.e., has been assigned k jobs) there are only a constant number of groups and, in particular, the cardinalities of the job sets assigned to any two machines differ only by a constant. We will also remove the assumption of knowing $p^{\infty}_{\max}$ in advance.

The algorithm. We first need to introduce some formal definitions. We think of every machine as having k slots, each of which may contain one job or it can be empty. We form k rows of these slots, where every row contains one slot of each machine. A row is *full* if each slot of the row has a job, and it is *not full* otherwise, i.e., at least one slot of the row is empty. A row is *free* if it has no job at all, i.e., all its slots are empty. In the following, let $p_{\max}$ denote the maximum job size of a job seen so far, and let $p^{\infty}_{\max}$ denote the maximum job size by the end of the instance. We keep track of $p_{\max}$ in an online fashion in the sense that whenever a new job is released, we check if we need to update (increase) these values.

We form a partition of the jobs (released so far) into the *groups* $G_0, \dots, G_\ell, G_\infty$, where $\ell = \lfloor 2\log(k) \rfloor$ and G_i, $i = 0, \dots, \ell$, contains all jobs j of size $p_j = p_{\max}/2^i$. Group G_∞ contains all other (smaller) jobs.

The algorithm is defined recursively and in the recursive calls, it may remove full rows from the existing schedule. The removal of the rows is only with respect to the rules of assigning future jobs to the machines (in the sense that the jobs that were assigned to slots of these rows are still assigned to the corresponding machines and are not actually removed). When rows are removed, and the number of rows is decreased, the value of k will be decreased and the value of ℓ will be updated accordingly. For a fixed value of ℓ, we can maintain the partition into groups in an online fashion. The value of ℓ may become smaller, in which case some groups are merged into the group of small jobs. Since the jobs of each group except for G_∞ share a common size, groups that are not merged remain unchanged.

To cope with the dynamic grouping where large jobs may later become small over time, we change the previous algorithm in the following way. Instead of keeping one row that is currently being filled for each group, we keep two. One of these two rows may also contain jobs from G_∞. To make this more precise, we now formally state the structural invariants of the algorithm's schedule.

The invariants of the algorithm. Consider the following structure in a schedule. For each $i = 0, \ldots, \ell$ there are exactly two rows r_i, r'_i containing elements of the group G_i. Row r_i contains only elements of G_i, whereas row r'_i contains elements of G_i and of G_∞. Moreover, of each pair of rows corresponding to a common group, at least one is not full. Finally, there are $\lceil k/2 \rceil - 2(\ell + 1)$ rows containing only elements from G_∞ and none of them is full. The remaining $\lfloor k/2 \rfloor$ rows are free.

The structure can be built trivially in the beginning when all rows are empty and $2(\ell+1) = 2\lfloor 2\log(k)\rfloor + 2 \leq \lceil k/2 \rceil$, which holds for all $k \geq 49$. In the case where k is initially smaller, the algorithm will output an arbitrary feasible schedule, which is 48-competitive.

The definition of this structure may give the (false) impression that the algorithm tries to keep the rows not full. This is not the case. In its recursive calls, the algorithm removes certain rows from the instance when they become full and the invariants above can also be read as: When two rows r_i, r'_i (or one row belonging to G_∞) become full, they need to be removed from the instance and new rows (e.g., from the empty ones) need to be allocated to take their place. When the algorithm removes a pair of rows (two rows corresponding to a common group become full), it also decreases k by 2.

We will argue inductively that given such a structure, we can assign the remaining jobs using the recursive algorithm in a way that maintains the invariants, and so that each machine ends up with a total load (including the jobs already in the schedule) of at most

$$\frac{2}{m} \cdot p(J) + 8 \sum_{i=\log(p_{\max})}^{\log(p^\infty_{\max})} 2^i + \left(42 - \frac{1}{k-1}\right) p^\infty_{\max}. \tag{1}$$

Notice that the second term is the sum of all job sizes (that is, all powers of 2) between $p_{\max}$ and $p^\infty_{\max}$ (multiplied with 8). Since the average load $p(J)/m$ and $1/2 \cdot \sum_{i=\log(p_{\max})}^{\log(p^\infty_{\max})} 2^i \leq p^\infty_{\max}$ form lower bounds on OPT, this constitutes a 60-competitive algorithm (120-competitive when taking into account the initial rounding). For $k \in \{48, 49\}$ it is trivial to see that there is an algorithm which (given the structure above) assigns all remaining jobs online while maintaining the bound (1) on the loads. This is because (1) is greater than $49p^\infty_{\max}$ and therefore any feasible schedule satisfies the bound. This proves the base case of our inductive argument.

The algorithm for scheduling the next job while maintaining the invariants and satisfying the load bound (1). Let h denote the number of free slots in our current schedule, that is, $k \cdot m$ minus the number of jobs in the schedule. Our induction is over k and h: If $k \in \{48, 49\}$, we observed that we can guarantee the makespan bound (1); the base case $h = 0$ does not have to be considered, since this contradicts the presumed structure on the current schedule (the induction will always end in $k \in \{48, 49\}$). Assume now that $k \geq 50$ and that we have an algorithm that for all $k' \in \{k, k-1, k-2\}$ and $h' < h$ can continue the schedule with the specified structure while guaranteeing the bound (1). For all $k \geq 50$ it holds that $2(\ell+1) = 2 + 2\lfloor 2\log(k)\rfloor < \lceil k/2 \rceil$. This implies that the number of rows dedicated to G_∞, $\lceil k/2\rceil - 2(\ell+1)$, is strictly positive and the number of free rows, $\lfloor k/2 \rfloor$, is at least 25, which will be important for our induction step. Suppose some job j_{new} arrives.

First consider the case that $p_{j_{\text{new}}} \leq p_{\max}$ (referring to the value of $p_{\max}$ before the arrival of j_{new}) and $j_{\text{new}} \notin G_\infty$. Let G_i be the group with $j_{\text{new}} \in G_i$. We assign j_{new} to an arbitrary empty slot in r_i or r'_i. If one of the rows remains not full, we have maintained the structure and use the induction hypothesis to prove that we can construct the remaining schedule with the desired guarantee. If on the other hand this makes both rows r_i and r'_i full, we first remove the two full rows and then we are going to use the induction hypothesis as described below: We set $k' := k - 2$. This reduces the index of the last group to $\ell' = \lfloor 2\log(k')\rfloor$. However, notice that $\ell \geq \ell' \geq \ell - 1$ (so it is possible that one group is merged into the group of small jobs).

Case 1: $\ell' = \ell$. We remove one row dedicated to G_∞ and pair it with an empty row to form new rows r'_i and r_i respectively (we recall that the numbers of such rows are positive before the removal). The number of rows dedicated to G_∞ is now $\lfloor k/2\rfloor - 2(\ell+1) - 1 = \lfloor k'/2\rfloor - 2(\ell'+1)$. Hence, the structure is repaired and we can use the induction hypothesis.

Case 2: $\ell' = \ell - 1$ and $i = \ell$. Then group G_i disappears, the number of rows dedicated to G_∞ is still $\lfloor k/2\rfloor - 2(\ell+1) = \lfloor k'/2\rfloor - 2(\ell'+1) - 1$. To repair the structure, we add one empty row to the group G_∞.

Case 3: $\ell' = \ell - 1$ and $i < \ell$. Then group G_ℓ is merged into G_∞, and it creates two new rows for G_∞ that were corresponding previously to G_ℓ. We create new rows r_i and r'_i corresponding to G_i by taking one of these rows previously corresponding to G_ℓ (the complete one if there is such a row) as r'_i and an empty one as r_i. This means the number of rows dedicated to G_∞ is $\lfloor k/2\rfloor - 2(\ell+1) + 1 = \lfloor k'/2\rfloor - 2(\ell'+1)$ and no row dedicated to G_∞ is full (because full rows of G_∞ will always be removed, and the only one that was perhaps temporarily added to the set of its rows was moved to the set of another group).

Let J_C denote the jobs in the two full rows that we have just removed. By induction hypothesis, we obtain a schedule for $J \setminus J_C$ where the load of each machine is at most

$$\frac{2}{m} \cdot p(J \setminus J_C) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k'-1}\right) p_{\max}^\infty .$$

Notice that the total size of the two jobs of J_C on each machine is at least $p_{\max}/2^i$ and at most $2 \cdot p_{\max}/2^i \leq 2/m \cdot p(J_C)$. Thus, the total load on each machine is at most

$$\frac{2}{m}p(J_C) + \frac{2}{m}p(J \setminus J_C) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k'-1}\right) p_{\max}^\infty \leq \frac{2}{m}p(J) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k-1}\right) p_{\max}^\infty .$$

Now consider the case that $j_{\text{new}} \in G_\infty$. We add j_{new} to an arbitrary row dedicated to G_∞. If the row remains not full, we can directly use the induction hypothesis as the structure is maintained. Otherwise, we remove the row and set $k' = k - 1$ (accordingly, $\ell' = \lfloor 2\log(k')\rfloor$). The number of rows dedicated to G_∞ has reduced to

$$\lfloor k/2\rfloor - 2(\ell+1) - 1 \leq \lfloor k'/2\rfloor - 2(\ell+1) \leq \lfloor k'/2\rfloor - 2(\ell'+1).$$

That means, we potentially have too few rows in G_∞, but not too many. On the other hand,

$$\lfloor k/2 \rfloor - 2(\ell+1) - 1 \geq \lfloor k'/2 \rfloor - 2(\ell'+2) - 1 = \lfloor k'/2 \rfloor - 2(\ell'+1) - 3.$$

So G_∞ is missing at most three rows. We fill up these missing row with empty ones (recall there are at least 25 empty rows) and we have maintained the required structure. Let again J_C denote the jobs in the full row. Using the induction hypothesis and that each job in J_C is of size less than $p_{\max}/k^2$ (by using the previous values of ℓ, this is the upper bound on the sizes of jobs of G_∞), we get a schedule with maximum load at most

$$\begin{aligned}
&\frac{2}{m}p(J \setminus J_C) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k'-1}\right) p_{\max}^\infty + \frac{1}{k^2} p_{\max} \\
&\leq \frac{2}{m}p(J) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k-2} + \frac{1}{k^2}\right) p_{\max}^\infty \\
&\leq \frac{2}{m}p(J) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k-1}\right) p_{\max}^\infty.
\end{aligned}$$

Finally consider the case that $p_{j_{\text{new}}} > p_{\max}$. This means the new maximal job size is $p'_{\max} = p_{j_{\text{new}}}$. From the job set J^0 of the jobs in our current schedule (not including jobs of removed rows) and excluding j_{new}, we construct a new job instance $J^{0'}$ where we increase the size of the jobs in G_i, $i = 0, \ldots, \ell$, from $p_{\max}/2^i$ to $p'_{\max}/2^i$ and we consider the same schedule for $J^{0'}$, which satisfies our required structure. Let J' be the union of jobs $J^{0'}$, j_{new}, and all remaining jobs that have not arrived yet. If we can assign all remaining jobs in this bigger instance J' with some bound on the loads of the machines, then we can in particular obtain the same bound on the maximum load for our original instance J. Notice that since in the current schedule there are only two rows containing jobs of each group G_i, $i < \infty$, there can be at most $2m$ jobs in total of each such group. Thus,

$$p(J') \leq p(J) + 2m \sum_{i=0}^{\ell} \frac{p'_{\max}}{2^i} \leq p(J) + 8m \cdot \frac{p'_{\max}}{2}.$$

In the previous two cases, we have already proved that (for the given numbers h and k) we can schedule the remaining jobs of J' with a bounded makespan, since we are in the case that $p_{j_{\text{new}}} \leq p'_{\max}$. More precisely, we obtain with the previous arguments a schedule for J' (and in particular for J) with a maximum load of at most

$$\frac{2}{m}p(J') + 8 \sum_{i=\log(p'_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k-1}\right) p_{\max}^\infty \leq \frac{2}{m}p(J) + 8 \sum_{i=\log(p_{\max})}^{\log(p_{\max}^\infty)} 2^i + \left(42 - \frac{1}{k-1}\right) p_{\max}^\infty,$$

where the inequality holds since $\log(p'_{\max}) \geq \log(p_{\max}) + 1$ due to the rounding. This concludes the proof.

▶ **Theorem 3.** *For cardinality constrained scheduling there is a* 120*-competitive online algorithm.*

3 Ordinal algorithms

A straight-forward approach in the ordinal setting is to assign the jobs via round-robin, i.e., the i-th largest job is assigned to machine $((i-1) \bmod m)+1$. Note that this already gives an ordinal algorithm of rate 2 and applies to both, the makespan minimization problem on identical machines as well as our problem. Thus, the goal of this section is to show that by delicately defining a different algorithm, we can get an ordinal algorithm of rate strictly smaller than 2.

Preliminaries and easy cases. We can always assume that the job sequence has $n = m \cdot k$ jobs. If it has more jobs then we can safely output that there is no feasible solution, and otherwise, we can add $n - m \cdot k$ jobs of size 0 at the end of the input sequence. These zero sized jobs do not change the optimal cost, and for every feasible solution of the original instance, there is a corresponding feasible solution of the same cost with the zero sizes jobs (by adding for each machine the number of jobs so that it will have exactly k jobs).

Furthermore, we will assume that $m \geq 2$ and $k \geq 3$. For one machine ($m = 1$), placing the k input jobs on the single given machine is a trivial ordinal algorithm with rate 1. Similarly, if $k = 1$, assigning the i-th largest of the m input jobs to machine i is again ordinal and optimal. Lastly, in the case $k = 2$, an optimal schedule can be achieved by assigning the first (largest) job to the first machine, the second to the second and so on until each machine received one job. Afterwards, we change the direction, that is, job $m + 1$ is assigned to machine m, job $m + 2$ to machine $m - 1$, and so forth.

First ideas. Observe that assigning the first m jobs to different machines is necessary in any algorithm of rate strictly smaller than 2 as the ordered input might consist of m jobs of size 1 and $m(k-1)$ jobs of size 0. On the other hand, the mentioned round-robin approach behaves badly if we have one big and many small jobs, e.g., if we have $m = k$, one job of size k, $k(k-1)$ jobs of size 1, and the remaining jobs of size 0. Then the first machine receives load $2k - 1$ in the round-robin approach while there is a trivial solution with objective value k.

Keeping these examples in mind, a first idea for an ordinal algorithm might be to spread out the first m jobs and afterward place fewer jobs on the machines that received the largest jobs. More concretely we could, for instance, place the first m jobs as described and then alternately take m and $\lceil m/2 \rceil$ jobs (from the input sequence, i.e., in non-increasing order) and place them on all and the last $\lceil m/2 \rceil$ machines, respectively. This is, in fact, the central idea for the known [30] ordinal algorithm with rate $\frac{5}{3}$ for the case without cardinality constraints. But in our case, we need another strategy after the last $\lceil m/2 \rceil$ machines each received k jobs. The most obvious idea at this point, would be to apply round robin to the remaining jobs and first $\lfloor m/2 \rfloor$ machines. However, it is relatively easy to see that we can adapt the bad example for round-robin to the resulting algorithm by simply doubling the number of machines and hence, a more sophisticated approach is needed.

Now, the first step of the algorithm presented here is again to place the i-th biggest job to machine i for $i \in [m]$. Then we use the above approach of placing jobs alternately on a smaller and a larger part of the last $\lceil m/2 \rceil$ machines and apply it repeatedly so that the machines are gradually filled up starting from the last machines. In the following, this approach is described in detail.

The algorithm for the general case. The assignment procedure works in *phases*, each of which is composed of *rounds*. Let $\xi = \lfloor \log m \rfloor + 2$ be the number of phases. A round is defined as an interval of consecutive machines $[m_\ell, m_r - 1]$, where the two indexes $m_\ell, m_r \in [m + 1]$

are called *border machines* (the value $m+1$ is to allow that an interval ends with the last machine). In a round defined by the interval $[m_\ell, m_r - 1]$, we assign the next $m_r - m_\ell$ jobs to the machines of this interval where the i-th largest job in this set of jobs that we assign in the round is assigned to machine $m_\ell + i - 1$.

Next, we define a subset of the machines that are border machines of some round of the algorithm. We set $\mu(i) = \lfloor \frac{m}{2^{\xi - i}} \rfloor + 1$ for each $i \in [\xi]$. The border machines of the rounds are always from the set $\{\mu(b) \mid b \in [\xi]\}$. Note that $\mu(1) = 1$, $\mu(2) = 2$ and $\mu(\xi) = m+1$ for any value of m. Observe that the difference between two consecutive border machines, i.e., $\mu(i+1) - \mu(i)$, grows approximately as a geometric sequence. We briefly consider some examples:

- If $m = \{2, 3\}$, we have $\xi = 3$ and the three borders are 1, 2, and 3 or 4 respectively.
- If $m \in \{4, 5, 6, 7\}$, we have $\xi = 4$, and the third border is 3 for $m = 4, 5$ and 4 for $m = 6, 7$.
- If $m = 2^q$ for some integer $q \geq 1$, we have $q + 2$ phases and $\{1\} \cup \{1 + 2^i \mid i \in \{0, \dots, q\}\}$ is the set of border machines.

We conclude that in order to define the assignment, we need to define the intervals of the rounds of every phase. Like in the last chapter, we use the intuition that each machine has k slots to be filled by jobs. Our algorithm works as follows, due to space restrictions, the proof is excluded and can be found in the full version, see [13]:

1. In the first phase, there is only one round with borders $\mu(1) = 1$ and $\mu(\xi) = m+1$.
2. In the second phase, we repeat the following until all the slots between $\mu(\xi - 1)$ and $\mu(\xi)$ are filled: One round with borders $\mu(\xi - 2)$ and $\mu(\xi)$, followed by two rounds with borders $\mu(\xi - 1)$ and $\mu(\xi)$ (or less rounds if each machine of this last interval has exactly k jobs).
3. In phase $s \in \{3, \dots, \xi - 1\}$, there are alternating rounds with borders $\mu(\xi - s)$, $\mu(\xi - s + 2)$ and $\mu(\xi - s + 1)$, $\mu(\xi - s + 2)$ respectively. There are as many rounds as are needed to fill all the slots of the interval between $\mu(\xi - s + 1)$ and $\mu(\xi - s + 2)$.
4. In the last phase, each round has borders $\mu(1) = 1$ and $\mu(2) = 2$, so in each such round we assign one job to machine 1 (and no other jobs to other machines). The number of rounds of this phase is so that the resulting number of jobs assigned to machine 1 in all phases is exactly k.

▶ **Theorem 4.** *There is an ordinal algorithm for cardinality constrained scheduling of rate at most* $\frac{81}{41}$.

4 Algorithms with constant migration factors

In this section, we consider algorithms with constant migration factor. Due to space restrictions, most of our results in this context are not included in this extended abstract. In the following, we focus on the connection between ordinal and robust algorithms.

▶ **Theorem 5.** *Given a polynomial time algorithm* ALG *for the ordinal settings of rate at most α, there is a robust $((1+\varepsilon)\alpha)$-approximation algorithm whose migration factor is* $\frac{1+\varepsilon}{\varepsilon}$.

Proof. Upon the release of a new job j, we immediately round up its size p_j to the next integer power of $(1+\varepsilon)$. Let p'_j be the rounded size of job j. Our algorithm ignores the original sizes of jobs and simply schedules the jobs of this rounded input. Observe that every feasible solution of the original instance is also a feasible solution of the rounded instance and vice-versa. Furthermore, the cost of a solution in terms of the original instance is at most its cost in terms of the rounded instance, and this last term is again at most $(1+\varepsilon)$ times the cost of the solution with respect to the original instance. Regarding the migration factor, there may be a multiplicative increase by a factor of $1+\varepsilon$. Thus, in order to prove the claim, it suffices to present an α-approximation algorithm for the rounded instance whose migration factor is at most $\frac{1}{\varepsilon}$. Thus, in the remainder of this proof we consider the rounded instance.

Our algorithm maintains a list of the jobs, ordered non-increasingly, that were already released followed by a sequence of jobs of size 0. The (already) released jobs are sorted in a non-decreasing order of their sizes. Based on this ordered list of jobs, we assign the jobs using the ordinal algorithm. The approximation ratio of the resulting algorithm is at most α (for the rounded input) based on the assumption on the rate of the ordinal algorithm. Thus, in order to prove the claim it suffices to show that we can maintain this sorted list (and its corresponding schedule) by migrating at most one job of each size that is smaller than the size of the newly arrived job. To see that, recall that in the rounded instance, all job sizes are integer powers of $1+\varepsilon$, and so the total size of migrated jobs when j is released would be at most $p'_j \cdot \sum_{i=1}^{\infty} \frac{1}{(1+\varepsilon)^i} = p'_j \cdot \frac{1}{\varepsilon}$ as we claimed.

In the rounded instance, we let a *size class* be the set of jobs of a common size, and this appears as a consecutive sublist of jobs in the sorted list. Observe that we can modify the sorted order of jobs by changing the order of jobs of a common size class (but when reflecting this change to the schedule this may create further migration). We append the new job j as the smallest job of its size class. Hence, it will be placed at the position of the largest job of the next smallest non-empty size class. We can remove and reinsert this job treating it the same way as we did the new job. Hence, for every non-empty size class whose common size is smaller than p'_j we take its largest job j' and move it to become the smallest of its size class. As the last step of this procedure, one of the size 0 dummy jobs is removed from the list. Observe that when reflected to the schedule, the jobs which were not the largest among their size class were not migrated by this resorting of the jobs. Furthermore only one job of each such size class is migrated. The running time of this procedure is linear (for every arriving job), so the claim follows. ◀

References

1 Susanne Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999.

2 Luitpold Babel, Bo Chen, Hans Kellerer, and Vladimir Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004.

3 János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. Online bin packing with cardinality constraints resolved. *Journal of Computer and System Sciences*, 112:34–49, 2020.

4 Nikhil Bansal, Tim Oosterwijk, Tjark Vredeveld, and Ruben van der Zwaan. Approximating vector scheduling: Almost matching upper and lower bounds. *Algorithmica*, 76(4):1077–1096, 2016.

5 József Békési, György Dósa, and Leah Epstein. Bounds for online bin packing with cardinality constraints. *Information and Computation*, 249:190–204, 2016.

6 Sebastian Berndt, Leah Epstein, Klaus Jansen, Asaf Levin, Marten Maack, and Lars Rohwedder. Online bin covering with limited migration. In *Proc. of the 27th Annual European Symposium on Algorithms, ESA 2019*, volume 144, pages 18:1–18:14, 2019.

7 Sebastian Berndt, Klaus Jansen, and Kim-Manuel Klein. Fully dynamic bin packing revisited. *Mathematical Programming*, 179(1):109–155, 2020.

8 Lin Chen, Klaus Jansen, Wenchang Luo, and Guochuan Zhang. An efficient PTAS for parallel machine scheduling with capacity constraints. In *International Conference on Combinatorial Optimization and Applications*, pages 608–623. Springer, 2016.

9 Mauro Dell'Amico and Silvano Martello. Bounds for the cardinality constrained $P||C_{max}$ problem. *Journal of Scheduling*, 4(3):123–138, 2001.

10 Mauro Dell'Amico, Manuel Iori, Silvano Martello, and Michele Monaci. Lower bounds and heuristic algorithms for the k_i-partitioning problem. *European Journal of Operational Research*, 171(3):725–742, 2006.

11 L. Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20(4):1015–1030, 2006.

12 Leah Epstein. A survey on makespan minimization in semi-online environments. *Journal of Scheduling*, 21(3):269–284, 2018.

13 Leah Epstein, Alexandra Lassota, Asaf Levin, Marten Maack, and Lars Rohwedder. Cardinality constrained scheduling in online models, 2022. `arXiv:2201.05113`.

14 Leah Epstein and Asaf Levin. A robust APTAS for the classical bin packing problem. *Mathematical Programming*, 119(1):33–49, 2009.

15 Leah Epstein and Asaf Levin. Robust approximation schemes for cube packing. *SIAM Journal on Optimization*, 23(2):1310–1343, 2013.

16 Leah Epstein and Asaf Levin. Robust algorithms for preemptive scheduling. *Algorithmica*, 69(1):26–57, 2014.

17 Leah Epstein and Asaf Levin. Robust algorithms for total completion time. *Discrete Optimization*, 33:70–86, 2019.

18 Björn Feldkord, Matthias Feldotto, Anupam Gupta, Guru Guruganesh, Amit Kumar, Sören Riechers, and David Wajc. Fully-dynamic bin packing with little repacking. In *Proc. of the 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, pages 51:1–51:24, 2018.

19 Rudolf Fleischer and Michaela Wahl. Online scheduling revisited. *Journal of Scheduling*, 3(6):343–353, 2000.

20 Waldo Gálvez, José A. Soto, and José Verschae. Symmetry exploitation for online machine covering with bounded migration. In *Proc. of the 26th European Symposium on Algorithms, ESA 2018*, pages 32:1–32:14, 2018.

21 Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

22 Yong He and Zhiyi Tan. Ordinal on-line scheduling for maximizing the minimum machine completion time. *Journal of Combinatorial Optimization*, 6(2):199–206, 2002.

23 Yong He, Zhiyi Tan, Jing Zhu, and Enyu Yao. k-partitioning problems for maximizing the minimum load. *Computers & Mathematics with Applications*, 46(10-11):1671–1681, 2003.

24 Klaus Jansen and Kim-Manuel Klein. A robust AFPTAS for online bin packing with polynomial migration. *SIAM Journal on Discrete Mathematics*, 33(4):2062–2091, 2019.

25 Klaus Jansen, Kim-Manuel Klein, Maria Kosche, and Leon Ladewig. Online strip packing with polynomial migration. In *Proc. of the 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2017*, pages 13:1–13:18, 2017.

26 Yasushi Kawase, Kei Kimura, Kazuhisa Makino, and Hanna Sumita. Optimal matroid partitioning problems. *Algorithmica*, 2021. To appear.

27 Hans Kellerer and Vladimir Kotov. A 3/2-approximation algorithm for k_i-partitioning. *Operations Research Letters*, 39(5):359–362, 2011.

28 K. L. Krause, V. Y. Shen, and H. D. Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the ACM*, 22(4):522–550, 1975.

29 Wei-Ping Liu and Jeffrey B Sidney. Bin packing using semi-ordinal data. *Operations research letters*, 19(3):101–104, 1996.

30 Wei-Ping Liu, Jeffrey B Sidney, and André Van Vliet. Ordinal algorithms for parallel machine scheduling. *Operations Research Letters*, 18(5):223–232, 1996.

31 John F. Rudin III. *Improved bounds for the on-line scheduling problem.* PhD thesis, The University of Texas at Dallas, 2001.

32 Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009.

33 Martin Skutella and José Verschae. Robust polynomial-time approximation schemes for parallel machine scheduling with job arrivals and departures. *Mathematics of Operations Research*, 41(3):991–1021, 2016.

34 Zhiyi Tan and Yong He. Semi-on-line scheduling with ordinal data on two uniform machines. *Operations Research Letters*, 28(5):221–231, 2001.

35 Zhiyi Tan, Yong He, and Leah Epstein. Optimal on-line algorithms for the uniform machine scheduling problem with ordinal data. *Information and Computation*, 196(1):57–70, 2005.

Detours in Directed Graphs

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway

Petr A. Golovach
Department of Informatics, University of Bergen, Norway

William Lochet
Department of Informatics, University of Bergen, Norway

Danil Sagunov
St. Petersburg Department of V.A. Steklov Institute of Mathematics, Russia
JetBrains Research, Saint Petersburg, Russia

Kirill Simonov
Algorithms and Complexity Group, TU Wien, Austria

Saket Saurabh
Institute of Mathematical Sciences, HBNI, Chennai, India
Department of Informatics, University of Bergen, Norway

Abstract

We study two “above guarantee” versions of the classical Longest Path problem on undirected and directed graphs and obtain the following results. In the first variant of Longest Path that we study, called Longest Detour, the task is to decide whether a graph has an (s, t)-path of length at least $\mathrm{dist}_G(s, t) + k$ (where $\mathrm{dist}_G(s, t)$ denotes the length of a shortest path from s to t). Bezáková et al. [7] proved that on undirected graphs the problem is fixed-parameter tractable (FPT) by providing an algorithm of running time $2^{\mathcal{O}(k)} \cdot n$. Further, they left the parameterized complexity of the problem on directed graphs open. Our first main result establishes a connection between Longest Detour on directed graphs and 3-Disjoint Paths on directed graphs. Using these new insights, we design a $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time algorithm for the problem on directed planar graphs. Further, the new approach yields a significantly faster FPT algorithm on undirected graphs.

In the second variant of Longest Path, namely Longest Path above Diameter, the task is to decide whether the graph has a path of length at least $\mathrm{diam}(G) + k$ ($\mathrm{diam}(G)$ denotes the length of a longest shortest path in a graph G). We obtain dichotomy results about Longest Path above Diameter on undirected and directed graphs. For (un)directed graphs, Longest Path above Diameter is NP-complete even for $k = 1$. However, if the input undirected graph is 2-connected, then the problem is FPT. On the other hand, for 2-connected directed graphs, we show that Longest Path above Diameter is solvable in polynomial time for each $k \in \{1, \ldots, 4\}$ and is NP-complete for every $k \geq 5$. The parameterized complexity of Longest Detour on general directed graphs remains an interesting open problem.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases longest path, longest detour, diameter, directed graphs, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.29

Related Version *Full Version*: https://arxiv.org/abs/2201.03318

Funding *Fedor V. Fomin*: supported by the Research Council of Norway via the project BWCA (grant no. 314528).
Petr A. Golovach: supported by the Research Council of Norway via the project BWCA (grant no. 314528).

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 29; pp. 29:1–29:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

William Lochet: supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416).
Danil Sagunov: supported by Leonhard Euler International Mathematical Institute in Saint Petersburg (agreement no. 075-15-2019-1620).
Kirill Simonov: supported by the Austrian Science Fund (FWF) via project Y1329 (Parameterized Analysis in Artificial Intelligence).
Saket Saurabh: supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416) and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

1 Introduction

In the Longest Path problem, we are given an n-vertex graph G and an integer k. (Graph G could be undirected or directed.) The task is to decide whether G contains a path of length at least k. Longest Path is a fundamental algorithmic problem that played one of the central roles in developing parameterized complexity [46, 9, 2, 36, 40, 13, 12, 41, 51, 26, 26, 25, 42, 8]. To further our algorithmic knowledge about the Longest Path problem, Bezáková et al. [7] introduced a novel "above guarantee" parameterization for the problem. For a pair of vertices s, t of an n-vertex graph G, let $\mathrm{dist}_G(s,t)$ be the distance from s to t, that is, the length of a shortest path from s to t. In this variant of Longest Path, the task is to decide whether a graph has an (s,t)-path of length at least $\mathrm{dist}_G(s,t)+k$. The difference with the "classical" parameterization is that instead of parameterizing by the path length, the parameterization is by the offset k.

Longest Detour **Parameter:** k
Input: A graph G, vertices $s, t \in V(G)$, and an integer k.
Task: Decide whether there is an (s,t)-path in G of length at least $\mathrm{dist}_G(s,t)+k$.

Since the length of a shortest path between s and t can be found in linear time, such a parameterization could provide significantly better solutions than parameterization by the path length. Bezáková et al. [7] proved that on undirected graphs the problem is fixed-parameter tractable (FPT) by providing an algorithm of running time $2^{\mathcal{O}(k)} \cdot n$. Parameterized complexity of Longest Detour on directed graphs was left as the main open problem in [7]. Our paper makes significant step towards finding a solution to this open problem.

Our results. Our first main result establishes a connection between Longest Detour and another fundamental algorithmic problem p-Disjoint Paths. Recall that the p-Disjoint Paths problem is to decide whether p pairs of *terminal* vertices (s_i, t_i), $i \in \{1, \ldots, p\}$, in a (directed) graph G could be connected by pairwise internally vertex disjoint (s_i, t_i)-paths. We prove (the formal statement of our result is given in Theorem 4) that if $\mathcal{C}$ is a class of (directed) graphs such that p-Disjoint Paths admits a polynomial time algorithm on $\mathcal{C}$ for $p = 3$, then Longest Detour is FPT on $\mathcal{C}$. Moreover, the FPT algorithm for Longest Detour on $\mathcal{C}$ is single-exponential in k (running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$).

Unfortunately, our result does not resolve the question about parameterized complexity of Longest Detour on directed graphs. Indeed, Fortune, Hopcroft, and Wyllie [29] proved that p-Disjoint Paths is NP-complete on directed graphs for every fixed $p \geq 2$. However, the new insight helps to establish the tractability of Longest Detour on planar directed graphs, whose complexity was also open. The theorem of Schrijver from [48] states that p-Disjoint Paths could be solved in time $n^{\mathcal{O}(p)}$ when the input is restricted to planar directed graphs. (This result was improved by Cygan et al. [17] who proved that p-Disjoint Paths parameterized by p is FPT on planar directed graphs.) Pipelined with our theorem, it immediately implies that Longest Detour is FPT on planar directed graphs.

Besides establishing parameterized complexity of Longest Detour on planar directed graphs our theorem has several advantages over the previous work even on undirected graphs. By the seminal result of Robertson and Seymour [47], p-Disjoint Paths is solvable in $f(p) \cdot n^3$ time on undirected graphs for some function f of p only. Therefore on undirected graphs p-Disjoint Paths is solvable in polynomial time for every fixed p, and for $p = 3$ in particular. Later the result of Robertson and Seymour was improved by Kawarabayashi, Kobayashi, and Reed [38] who gave an algorithm with quadratic dependence on the input size. Pipelined with our result, this brings us to a Monte Carlo randomized algorithm solving Longest Detour on undirected graphs in time $10.8^k \cdot n^{\mathcal{O}(1)}$. Our algorithm can be derandomized, and the deterministic algorithm runs in time $45.5^k \cdot n^{\mathcal{O}(1)}$. While the algorithm of Bezáková et al. [7] for undirected graphs runs in time $\mathcal{O}(c^k \cdot n)$, that is, is single-exponential in k, the constant c is huge. The reason is that their algorithm exploits the Win/Win approach based on excluding graph minors. More precisely, Bezáková et al. proved that if a 2-connected graph G contains as a minor, a graph obtained from the complete graph K_4 by replacing each edge by a path with k edges, then G has an (s,t)-path of length at least $\mathrm{dist}_G(s,t) + k$. Otherwise, in the absence of such a graph as a minor, the treewidth of G is at most $32k + 46$. Combining this fact with an FPT 3-approximation algorithm [11], running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, to compute the treewidth of a graph, brings us to a graph of treewidth at most $96k + \mathcal{O}(1)$. Finally, solving Longest Detour on graphs of bounded treewidth by one of the known single-exponential algorithms, see [18, 10, 27], will result in running time $3^{96k} \cdot n^{\mathcal{O}(1)}$. Thus on undirected graphs, our algorithm reduces the constant c in the base of the exponent from 3^{96} down to 10.8!

Our second set of results addresses the complexity of the problem strongly related to Longest Detour. The length of a longest shortest path in a graph G is denoted by *diameter of* G, $\mathrm{diam}(G)$. Thus every graph G has a path of length at least $\mathrm{diam}(G)$. But does it have a path of length longer than $\mathrm{diam}(G)$? This leads to the following parameterized problem.

Longest Path above Diameter **Parameter:** k
Input: A graph G and an integer k.
Task: Decide whether there is a path in G of length at least $\mathrm{diam}(G) + k$.

As in Longest Detour, the parameterization is by the offset k. When (s,t) is a pair of diametral vertices in G, the length of the shortest (s,t)-path in G is the diameter of G. However, this does not allow to reduce Longest Path above Diameter to Longest Detour– if there is a path of length $\mathrm{diam}(G) + k$ in G, it is not necessarily an (s,t)-path. Moreover, such a path might connect two vertices with a much smaller distance between them than $\mathrm{diam}(G)$. In fact, our hardness results for Longest Path above Diameter are based precisely on instances where the target path has this property: its length is very close to $\mathrm{diam}(G)$, but much larger than the shortest distance between its endpoints. Thus, the lower bounds we obtain for Longest Path above Diameter are not applicable to Longest Detour.

We obtain the following dichotomy results about Longest Path above Diameter on undirected and directed graphs. For undirected graphs, Longest Path above Diameter is NP-complete even for $k = 1$. However, if the input undirected graph is 2-connected, that is, it remains connected after deleting any of its vertices, then the problem is FPT. For directed graphs, the problem is also NP-complete even for $k = 1$. However, the situation is more complicated and interesting on 2-connected directed graphs. (Let us remind that a strongly connected digraph G is 2-connected or strongly 2-connected, if for every vertex $v \in V(G)$, graph $G - v$ remains strongly connected.) In this case, we show that Longest Path above Diameter is solvable in polynomial time for each $k \in \{1, \ldots, 4\}$ and is NP-complete for every $k \geq 5$.

Our approach. A natural way to approach Longest Detour on directed graphs would be to mimic the algorithm for undirected graphs. By the result of Kawarabayashi and Kreutzer [39], every directed graph of sufficiently large directed treewidth contains a sizable directed grid as a "butterfly minor". However, as reported in [6], there are several obstacles towards applying the grid theorem of Kawarabayashi and Kreutzer for obtaining a Win/Win algorithm. After several unsuccessful attempts, we switched to another strategy.

We start the proof of Theorem 4 by checking whether G has an (s,t)-path of length $\text{dist}_G(s,t) + \ell$ for $k \leq \ell < 2k$. This can be done in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ by calling the algorithm of Bezáková et al. [7] that finds an (s,t)-path in a directed G of length *exactly* $\text{dist}_G(s,t) + \ell$. If such a path is not found, we conclude that if (G,k) is a yes-instance, then G contains an (s,t)-path of length at least $\text{dist}_G(s,t) + 2k$.

Next, we check whether there exist two vertices v and w reachable from s such that $\text{dist}_G(s,w) - \text{dist}_G(s,v) \geq k$ and G has pairwise disjoint (s,w)-, (w,v)-, and (v,t)-paths. If such a pair of vertices exists, we obtain a solution by concatenating disjoint (s,w)-, (w,v)-, and (v,t)-paths. This is the place in our algorithm, where we require a subroutine solving 3-Disjoint Paths.

When none of the above procedures finds a detour, we prove a combinatorial claim that allows reducing the search of a solution to a significantly smaller region of the graph. This combinatorial claim is the essential part of our algorithm. More precisely, we show that there are two vertices u and x, and a specific induced subgraph H of G (depending on u and x) such that G has an (s,t)-path of length at least $\text{dist}_G(s,t) + k$ if and only if H has an (u,x)-path of length at least ℓ for a specific $\ell \leq 2k$ (also depending on u and x). Moreover, given u, in polynomial time, we can find a feasible domain for vertex x, and for each choice of x, we can also determine ℓ and construct H in polynomial time. Then we apply the algorithm of Fomin et al. [28] to check whether H has an (u,x)-path in H of length at least ℓ.

Our strategy for Longest Path above Diameter is different. For undirected graphs, the solution turns out to be reasonably simple. It easy to show that Longest Path above Diameter is NP-complete for $k = 1$ by reducing Hamiltonian Path to it. When an undirected graph G is 2-connected, and the diameter is larger than $k + 1$, then G always contains a path of length at least $d + k$. If the diameter is at most k, it suffices to run a Longest Path algorithm to show that the problem is FPT. For directed graphs, a similar reduction shows that the problem is NP-complete for $k = 1$. However, for 2-strongly-connected directed graphs, the situation is much more interesting. It is not too difficult to prove that when the diameter of a 2-strongly-connected digraph is sufficiently large, it always contains a path of length $\text{diam}(G) + 1$. With much more careful arguments, it is possible to push this up to $k = 4$. Thus for each $k \leq 4$, the problem is solvable in polynomial time. For $k = 5$ we can construct a family of 2-strongly-connected digraphs of arbitrarily large diameter that do not have a path of length $\text{diam}(G) + 5$. These graphs become extremely useful as gadgets that we use to prove that the problem is NP-complete for each $k \geq 5$.

Related work. There is a vast literature in the field of parameterized complexity devoted to Longest Path [46, 9, 2, 36, 40, 13, 12, 41, 51, 26, 26, 8]. The surveys [25, 42] and the textbook [16, Chapter 10] provide an overview of the advances in the area.

Longest Detour was introduced by Bezáková et al. in [7]. They gave an FPT algorithm for undirected graphs and posed the question about detours in directed graphs. Even the existence of a polynomial time algorithm for Longest Detour with $k = 1$, that is, deciding whether a directed graph has a path longer than a shortest (s, t)-path, is open. For the related Exact Detour problem, deciding whether there is a detour of length *exactly* $\text{dist}_G(s, t) + k$ is FPT both on directed and undirected graphs [7].

Another problem related to our work is Long (s, t)-Path. Here for vertices s and t of a graph G, and integer parameter k, we have to decide whether there is an (s, t)-path in G of length at least k. A simple trick, see [16, Exercise 5.8], allows to use color-coding to show that Long (s, t)-Path is FPT on undirected graph. For directed graphs the situation is more involved, and the first FPT algorithm for Long (s, t)-Path on directed graphs was obtained only recently [28]. The proof of Theorem 4 uses some of the ideas developed in [28].

Both Longest Detour and Longest Path above Diameter fit into the research subarea of parameterized complexity called “above guarantee” parameterization [44, 1, 15, 31, 32, 33, 34, 35, 43, 45]. Besides the work of Bezáková et al. [6], several papers study parameterization of longest paths and cycles above different guarantees. Fomin et al. [23] designed parameterized algorithms for computing paths and cycles longer than the girth of a graph. The same set of the authors in [22] studied FPT algorithms that finds paths and cycles above degeneracy. Fomin et al. [24] developed an FPT algorithm computing a cycle of length $2\delta + k$, where δ is the minimum vertex degree of the input graph. Jansen, Kozma, and Nederlof in [37] looked at parameterized complexity of Hamiltonicity below Dirac’s conditions. Berger, Seymour, and Spirkl in [5], gave a polynomial time algorithm that, with input a graph G and two vertices s, t of G, that decides whether there is an *induced* (s, t)-path that is longer than a shortest (s, t)-path. All these algorithms for computing long paths and cycles above some guarantee are for undirected graphs.

The remaining part of this paper is organized as follows. In Section 2, we give preliminaries. In Section 3, we prove our first main result establishing connections between 3-Disjoint Paths and Longest Detour (Theorem 4). Section 4 is devoted to Longest Path above Diameter. The concluding Section 5 provides open questions for further research.

2 Preliminaries

Parameterized Complexity. We refer to the recent books [16, 20] for the detailed introduction to Parameterized Complexity. Here we just remind that the computational complexity of an algorithm solving a parameterized problem is measured as a function of the input size n of a problem and an integer *parameter* k associated with the input. A parameterized problem is said to be *fixed-parameter tractable* (or FPT) if it can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function $f(\cdot)$.

Graphs. Recall that an undirected graph is a pair $G = (V, E)$, where V is a set of vertices and E is a set of unordered pairs $\{u, v\}$ of distinct vertices called *edges*. A directed graph $G = (V, A)$ is a pair, where V is a set of vertices and A is a set of ordered pairs (u, v) of distinct vertices called *arcs*. Note that we do not allow loops and multiple edges or arcs. We use $V(G)$ and $E(G)$ ($A(G)$, respectively) to denote the set of vertices and the set of edges (set of arcs, respectively) of G. We write n and m to denote the number of vertices and edges (arcs, respectively) if this does not create confusion. For a (directed) graph G and a subset $X \subseteq V(G)$ of vertices, we write $G[X]$ to denote the subgraph of G induced by X. For a set of vertices S, $G - S$ denotes the (directed) graph obtained by deleting the vertices of S, that is, $G - S = G[V(G) \setminus S]$. We write $P = v_1 \cdots v_k$ to denote a *path* with the vertices $v_1, \ldots, v_k$ and the edges $\{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$ (arcs $(v_1, v_2), \ldots, (v_{k-1}, v_k)$, respectively); v_1 and v_k are the *end-vertices* of P and the vertices $v_2, \ldots, v_{k-1}$ are *internal.* We say that P is an (v_1, v_k)*-path.* The *length* of P, denoted by $\text{length}(P)$, is the number of edges (arcs, respectively) in P. Two paths are *disjoint* if they have no common vertex and they are *internally disjoint* if no internal vertex of one path is a vertex of the other. For a (u, v)-path P_1 and a (v, w)-path P_2 that are internally disjoint, we denote by $P_1 \circ P_2$ the *concatenation* of P_1 and P_2. A vertex v is *reachable* from a vertex u in a (directed) graph G if G has a (u, v)-path. For $u, v \in V(G)$, $\text{dist}_G(u, v)$ denotes the *distance* between u and v in G, that is, the minimum number of edges (arcs, respectively) in an (u, v)-path. An undirected graph G is *connected* if for every two vertices u and v, G has a (u, v)-path. A directed graph G is *strongly-connected* if for every two vertices u and v both u is reachable form v and v is reachable from u. For a positive integer k, an undirected (directed, respectively) graph G is k*-connected* (k*-strongly-connected*, respectively) if $|V(G)| \geq k$ and $G - S$ is connected (strongly-connected, respectively) for every $S \subseteq V(G)$ of size at most $k - 1$. For a directed graph G, by G^T we denote the *transpose* of G, i.e. G^T is a directed graph defined on the same set of vertices and the same set of arcs, but the direction of each arc in G^T is reversed.

We use several known parameterized algorithms for finding long paths. First of all, let us recall the currently fastest deterministic algortihm for Longest Path on directed graphs due to Tsur [50].

▶ **Proposition 1** ([50]). *There is a deterministic algorithm for Longest Path with running time* $2.554^k \cdot n^{\mathcal{O}(1)}$.

We also need the result of Fomin et al. [28] for the Long Directed (s, t)-Path problem. This problem asks, given a directed graph G, two vertices $s, t \in V(G)$, and an integer $k \geq 0$, whether G has an (s, t)-path of length at least k.

▶ **Proposition 2** ([28]). *Long Directed* (s, t)*-Path can be deterministically solved in time* $4.884^k \cdot n^{\mathcal{O}(1)}$.

Clearly, both results holds for the variant of the problem on undirected graphs.

Finally, we use the result of Bezáková et al. [7] for the variant of Longest Detour whose task is, given a (directed) graph G, two vertices $s, t \in V(G)$, and an integer $k \geq 0$, decide whether G has an (s, t)-path of length *exactly* $\text{dist}_G(s, t) + k$.

▶ **Proposition 3** ([7]). *There is a bounded-error randomized algorithm that solves Exact Detour on undirected graphs in time* $2.746^k \cdot n^{\mathcal{O}(1)}$ *and on directed graphs in time* $4^k \cdot n^{\mathcal{O}(1)}$. *For both undirected and directed graphs, there is a deterministic algorithm that runs in time* $6.745^k \cdot n^{\mathcal{O}(1)}$.

3 An FPT algorithm for finding detours

In this section we show the first main result of our paper.

▶ **Theorem 4.** *Let* $\mathcal{C}$ *be a class of directed graphs such that* 3*-Disjoint Paths can be solved in* $f(n)$ *time time on* $\mathcal{C}$. *Then Longest Detour can be solved in* $45.5^k \cdot n^{\mathcal{O}(1)} + \mathcal{O}(f(n)n^2)$ *time by a deterministic algorithm and in* $23.86^k \cdot n^{\mathcal{O}(1)} + \mathcal{O}(f(n)n^2)$ *time by a bounded-error randomized algorithm when the input is restricted to graphs from* $\mathcal{C}$.

Proof. Let (G, s, t, k) be an instance of Longest Detour with $G \in \mathcal{C}$. For $k = 0$, the problem is trivial and we assume that $k \geq 1$. We also have that (G, s, t, k) is a trivial no-instance if t is not reachable from s. We assume from now that every vertex of G is reachable from s. Otherwise, we set $G := G[R]$, where R is the set of vertices of G reachable from s using the straightforward property that every (s,t)-path in G is a path in $G[R]$. Clearly, R can be constructed in $\mathcal{O}(n + m)$ time by the breadth-first search.

Using Proposition 3, we check in $6.745^{2k} \cdot n^{\mathcal{O}(1)}$ time by a deterministic algorithm (in $4^{2k} \cdot n^{\mathcal{O}(1)}$ time by a randomized algorithm, respectively) whether G has an (s,t)-path of length $\text{dist}_G(s,t) + \ell$ for some $k \leq \ell \leq 2k - 1$ by trying all values of ℓ in this interval. We return a solution and stop if we discover such a path. Assume from now that this is not the case, that is, if (G, s, t) is a yes-instance, then the length of every (s,t)-path of length at least $\text{dist}_G(s,t) + k$ is at least $\text{dist}_G(s,t) + 2k$.

We perform the breadth-first search from s in G. For an integer $i \geq 0$, denote by L_i the set of vertices at distance i from s. Let ℓ be the maximum index such that $L_\ell \neq \emptyset$. Because every vertex of G is reachable from s, $V(G) = \bigcup_{i=0}^{\ell} L_i$. We call $L_0, \ldots, L_\ell$ *BFS-levels.*

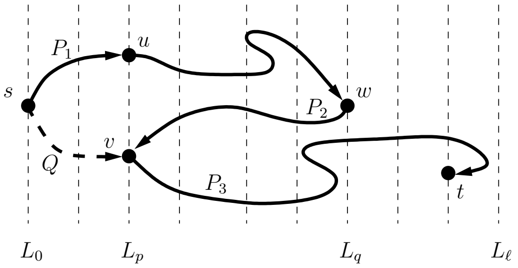


Figure 1 The choice of the BFS-levels L_p and L_q, vertices u, v, and w, and the paths P_1, P_2, and P_3.

Our algorithm is based on structural properties of potential solutions. Suppose that (G, s, t, k) is a yes-instance and let a path P be a solution of minimum length, that is, P is an (s,t)-path of length at least $\text{dist}_G(s,t) + k$ and among such paths the length of P is minimum. Denote by $p \in \{1, \ldots, \ell\}$ the minimum index such that L_p contains at least two vertices of G. Such an index exists, because if $|V(P) \cap L_i| \leq 1$ for all $i \in \{1, \ldots, \ell\}$, then P is a shortest (s,t)-path by the definition of $L_0, \ldots, L_\ell$ and the length of P is $\text{dist}_G(s,t) < \text{dist}_G(s,t) + k$ as $k \geq 1$. Let u be the first (in the path order) vertex of P in L_p and let $v \neq u$ be the second vertex of P that occurs in L_p. Denote by P_1, P_2, and P_3 the (s,u), (u,v), and (v,t)-subpath of P, respectively. Clearly, $P = P_1 \circ P_2 \circ P_3$. Let $q \in \{p, \ldots, \ell\}$ be the maximum index such that P_2 contains a vertex of L_q. Then denote by w the first vertex of P_2 in L_q. See Figure 1 for the illustration of the described configuration. We use this notation for a (hypothetical) solution throughout the proof of the theorem. The following claim is crucial for us.

▷ Claim 5. The length of P_2 is at least k.

Proof of Claim 5. For the sake of contradiction, assume that the length of P_2 is less than k. Let Q be a shortest (s,v)-path in G. By the definition of BFS-levels, $V(Q) \subseteq L_0 \cup \cdots \cup L_p$ and v is a unique vertex of Q in L_p. This implies that Q is internally vertex disjoint with P_3. Note that the length of Q is the same as the length of P_1, because P_1 contains exactly one vertex from each of the BFS levels $L_1, \ldots, L_p$. Then $P' = Q \circ P_3$ is an (s,t)-path and

$$\begin{aligned}\text{length}(P') =& \text{length}(Q) + \text{length}(P_3) = \text{length}(P_1) + \text{length}(P_3)\\ =& \text{length}(P) - \text{length}(P_2) \leq \text{length}(P) - k.\end{aligned}$$

Recall that the length of every (s,t)-path of length at least $\mathrm{dist}_G(s,t)+k$ is at least $\mathrm{dist}_G(s,t)+2k$. This means that $\mathrm{length}(P) \geq \mathrm{dist}_G(s,t)+2k$ and, therefore, the length of P' is at least $\mathrm{dist}_G(s,t)+k$, that is, P' is a solution to the considered instance. However, $\mathrm{length}(P') < \mathrm{length}(P)$, because P_2 contains at least one arc. This contradicts the choice of P as a solution of minimum length. This completes the proof of the claim. ◁

By Claim 5, solving Longest Detour on (G,s,t,k) boils down to identifying internally disjoint P_1, P_2, and P_3, where the length of P_2 is at least k.

First, we check whether we can find paths for $q-p \geq k-1$. Notice that if $q-p \geq k-1$, then for every internally disjoint (s,w)-, (w,v)-, and (v,t)-paths R_1, R_2, and R_3 respectively, their concatenation $R_1 \circ R_2 \circ R_3$ is an (s,t)-path of length at least $\mathrm{dist}_G(s,t)+k$. Recall that $G \in \mathcal{C}$ and p-Disjoint Paths can be solved in polynomial time on this graph class for $p=3$. For every choice of two vertices $w,v \in V(G)$, we solve p-Disjoint Paths on the instance $(G,(s,w),(w,v),(v,s))$. Then if there are paths R_1, R_2, and R_3 forming a solution to this instance, we check whether $\mathrm{length}(R_1)+\mathrm{length}(R_2)+\mathrm{length}(R_3) \geq \mathrm{dist}_G(s,t)+k$. If this holds, we conclude that the path $R_1 \circ R_2 \circ R_3$ is a solution to the instance (G,s,t,k) of Longest Detour and return it. Assume from now that this is not the case, that is, we failed to find a solution of this type. Then we can complement Claim 5 by the following observation about our hypothetical solution P.

▷ Claim 6. $q-p \leq k-2$.

This means that we can assume that $k \geq 2$ and have to check whether we can identify P_1, P_2, and P_3, where $V(P_2) \subseteq \bigcup_{i=p}^{p+k-2} L_i$. For this, we go over all possible choices of u. Note that the choice of u determines p, i.e., the index of the BFS-level containing u. We consider the following two cases for each considered choice of u.

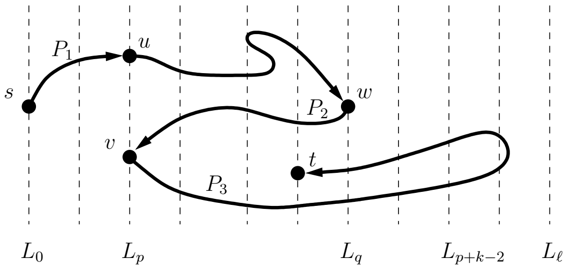


Figure 2 The structure of paths P_1, P_2, and P_3 in Case 1.

Case 1. $t \in L_r$ for some $p \leq r \leq p+k-2$ (see Figure 2). Then $\mathrm{dist}_G(s,t)=r$ and (G,s,t,k) is a yes-instance if and only if $G[L_p \cup \cdots \cup L_\ell]$ has a (u,t)-path S of length at least $(r-p)+k$, because the (s,u)-subpath of a potential solution should be a shortest (s,u)-path. Since $r-p \leq k-2$, we have that $(r-p)+k \leq 2k-2$ and we can find S in $4.884^{2k} \cdot n^{\mathcal{O}(1)}$ time by Proposition 2 if it exists. If we obtain S, then we consider an arbitrary shortest (s,u)-path S' in G and conclude that $S' \circ S$ is a solution. This completes Case 1.

Case 2. $t \in L_r$ for some $r \geq p+k-1$ (see Figure 3). We again consider our hypothetical solution $P = P_1 \circ P_2 \circ P_3$. Let $H = G[L_{p+k-1} \cup \cdots \cup L_\ell]$. Denote by X the set of vertices $x \in V(H)$ such that t is reachable from x in H. Denote by x the first vertex of P_3 in X. Clearly, such a vertex exists because $t \in X$. Moreover, $x \in L_{p+k-1}$ and its predecessor y in P_3 is in L_{p+k-2}. Otherwise, t would be reachable from $y \in V(H)$ in H contradicting the choice of x. Let Q_1 and Q_2 be the (v,y)- and (x,t)-subpaths of P_3. Then $P_3 = Q_1 \circ yx \circ Q_2$. We show one more claim about the hypothetical solution P.

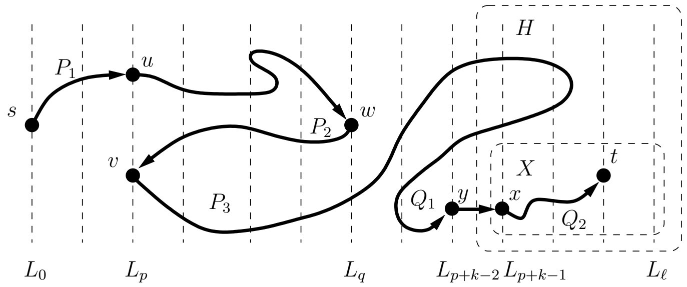


Figure 3 The structure of paths P_1, P_2, and P_3 in Case 2.

▷ Claim 7. $V(Q_1) \cap X = \emptyset$.

Proof of Claim 7. The proof is by contradiction. Assume that $z \in V(Q_1) \cap X$. Then t is reachable from z in H. However, x is the first vertex of P_3 with this property by the definition; a contradiction. ◁

Notice that because $x \in X$, there is an (x,t)-path Q_2' with $V(Q_2') \subseteq X$. By Claim 7, Q_1 and Q_2' are disjoint. Since $X \subseteq L_{p+k-1} \cup \dots \cup L_\ell$, we have that $(V(P_1) \cup V(P_2)) \cap X = \emptyset$. In particular, Q_2' is disjoint with P_1 and P_2 as well. Let $P_3' = Q_1 \circ yx \circ Q_2'$. By Claim 5, $P' = P_1 \circ P_2 \circ P_3'$ is a solution, because $\text{length}(P_2) \geq k$. This allows us to conclude that (G,s,t,k) has a solution (for the considered choice of u) if and only if there is $y \in L_{p+k-2}$ such that

(i) there is $x \in X$ such that $(y,x) \in A(G)$, and
(ii) the graph $G[L_p \cup \dots \cup L_\ell] - X$ has a (u,y)-path of length at least $2k-2$.

Our algorithm proceeds as follows. We construct the set X using the breadth-first search in $\mathcal{O}(n+m)$ time. Then for every $y \in L_{p+k-2}$ we check (i) whether there is $x \in X$ such that $(y,x) \in A(G)$, and (ii) whether $G[L_p \cup \dots \cup L_\ell] - X$ has a (u,y)-path S of length at least $2k-2$. To verify (ii), we apply Proposition 2 that allows to perform the check in $4.884^{2k} \cdot n^{\mathcal{O}(1)}$ time. If we find such a vertex y and path S, then to obtain a solution, we consider an arbitrary shortest (s,u)-path S' and an arbitrary (x,t) path S'' in $G[X]$. Then $P' = S' \circ S \circ yx \circ S''$ is a required solution to (G,s,t,k). This concludes the analysis in Case 2 and the construction of the algorithm.

The correctness of our algorithm has been proved simultaneously with its construction. The remaining task is to evaluate the total running time. Recall that we verify in $6.745^{2k} \cdot n^{\mathcal{O}(1)}$ time whether G has an (s,t)-path of length $\text{dist}_G(s,t) + \ell$ for some $k \leq \ell \leq 2k-1$ by a deterministic algorithm, and we need $4^{2k} \cdot n^{\mathcal{O}(1)}$ time if we use a randomized algorithm. Then we construct the BFS-levels in linear time. Next, we consider $\mathcal{O}(n^2)$ choices of v and w and apply the algorithm for 3-Disjoint Paths $(G,(s,w),(w,v),(v,s))$ in $f(n)$ time. If we failed to find a solution so far, we proceed with $\mathcal{O}(n)$ possible choices of u and consider either Case 1 or 2 for each choice. In Case 1, we solve the problem in $4.884^{2k} \cdot n^{\mathcal{O}(1)}$ time. In Case 2, we construct X in $\mathcal{O}(n+m)$ time. Then for $\mathcal{O}(n)$ choices of y, we verify conditions (i) and (ii) in $4.884^{2k} \cdot n^{\mathcal{O}(1)}$ time. Summarizing, we obtain that the total running time is $6.745^{2k} \cdot n^{\mathcal{O}(1)} + \mathcal{O}(f(n)n^2)$. Because $6.745^2 < 45.5$, we have that the deterministic algorithm runs in $45.5^k \cdot n^{\mathcal{O}(1)} + \mathcal{O}(f(n)n^2)$ time. Since $4^2 < 4.884^2 < 23.86$, we conclude that the problem can be solved in $23.86^k \cdot n^{\mathcal{O}(1)} + \mathcal{O}(f(n)n^2)$ time by a bounded-error randomized algorithm. ◀

In particular, combining Theorem 4 with the results of Cygan et al. [17], we obtain the following corollary.

▶ **Corollary 8.** *Longest Detour can be solved in $45.5^k \cdot n^{\mathcal{O}(1)}$ time by a deterministic algorithm and in $23.86^k \cdot n^{\mathcal{O}(1)}$ time by a bounded-error randomized algorithm on planar directed graphs.*

Using the fact that p-Disjoint Paths can be solved in $\mathcal{O}(n^2)$ time by the results of Kawarabayashi, Kobayashi, and Reed [38], we immediately obtain the result for Longest Detour on undirected graphs. However, we can improve the running time of a randomized algorithm by tuning our algorithm for the undirected case.

▶ **Corollary 9.** *Longest Detour can be solved in $45.5^k \cdot n^{\mathcal{O}(1)}$ time by a deterministic algorithm and in $10.8^k \cdot n^{\mathcal{O}(1)}$ time by a bounded-error randomized algorithm on undirected graphs.*

Proof. The deterministic algorithm is the same as in the directed case. To obtain a better randomized algorithm, we follow the algorithm from Theorem 4 and use the notation introduced in its proof. Let (G, s, t, k) be an instance of Longest Detour with $G \in \mathcal{C}$. We assume without loss of generality that $k \geq 1$ and G is connected. Using Proposition 3, we check in $2.746^{2k} \cdot n^{\mathcal{O}(1)}$ time by a randomized algorithm whether G has an (s,t)-path of length $\text{dist}_G(s,t) + \ell$ for some $k \leq \ell \leq 2k-1$. If we fail to find a solution this way, we construct the BFS-levels $L_0, \dots, L_\ell$.

Suppose that (G, s, t, k) is a yes-instance with a hypothetical solution P composed by the concatenation of P_1, P_2, and P_3 as in the proof of Theorem 4. Let also L_p and L_q be the corresponding BFS-levels. Observe that if $q - p \geq k/2$, then $\text{length}(P_2) \geq k$, because for every edge $\{x, y\}$ of G, x and y are either in the same BFS-level or in consecutive levels contrary to the directed case where we may have an arc (x, y) where $x \in L_i$ and $y \in L_j$ for arbitrary $j \in \{0, \dots, i\}$. Recall that for every choice of two vertices $w, v \in V(G)$, we solve p-Disjoint Paths on the instance $(G, (s, w), (w, v), (v, s))$ and try to find a solution to (G, s, t, k) by concatenating the solutions for these instances of p-Disjoint Paths. If we fail to find a solution this way, we can conclude now that $q - p \leq k/2 - 1$ improving Claim 6. Further, we pick u and consider two cases.

In Case 1, where $t \in L_r$ for some $p \leq r \leq p + k/2 - 1$, we now find a (u, t)-path S in $G[L_p \cup \dots \cup L_\ell]$ of length at least $(r - p) + k \leq 3k/2$ in $4.884^{3k/2} \cdot n^{\mathcal{O}(1)}$ time. If such a path exists, we obtain a solution.

In Case 2, where $t \in L_r$ for some $r \geq p + k/2$, we consider $H = G[L_{h+1} \cup \dots \cup L_\ell]$ for $h = p + \lceil k/2 \rceil$ and denote by X the set of vertices of the connected component of H containing X. Then for every $y \in L_h$ we check (i) whether there is $x \in X$ such that $\{y, x\} \in E(G)$, and (ii) whether $G[L_p \cup \dots \cup L_\ell] - X$ has a (u, y)-path S of length at least $k + \lceil k/2 \rceil$ in $4.884^{3k/2} \cdot n^{\mathcal{O}(1)}$ time. If such a path exists, we construct a solution containing it in the same way as on the directed case.

The running time analysis is essentially the same as in the proof of Theorem 4. The difference is that now we have that $2.746^2 \leq 4.884^{3/2} < 10.80$. This implies that the algorithm runs in $10.8^k \cdot n^{\mathcal{O}(1)}$ time. ◀

4 Longest Path Above Diameter

In this section, we investigate the complexity of Longest Path above Diameter. It can be noted that this problem is NP-complete in general even for $k = 1$.

▶ **Proposition 10.** *Longest Path above Diameter is* NP*-complete for $k = 1$ on undirected graphs.*

Proof. Let G be an undirected graph with $n \geq 2$ vertices. We construct the graph G' as follows (see Figure 4).

- Construct a copy of G.
- Construct a vertex u and make it adjacent to every vertex of the copy of G.
- Construct two vertices s and t, and then (s,u) and (u,t) paths P_s and P_t, respectively, of length $n-1$.

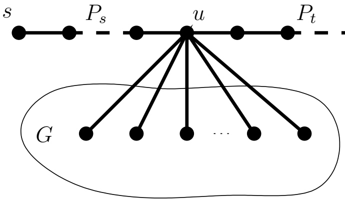


Figure 4 Construction of G'.

Notice that $\text{diam}(G) = \text{length}(P_s) + \text{length}(P_t) = 2n-2$. It is easy to verify that G' has a path of length $2n-1$ if and only if G has a path of length $n-1$, that is, G is Hamiltonian. Because Hamiltonian Path is well-known to be NP-complete [30], we conclude that Longest Path above Diameter is NP-complete for $k = 1$ ◀

Proposition 10 immediately implies that Longest Path above Diameter is NP-complete for $k = 1$ on strongly connected directed graphs as we can reduce the problem on undirected graphs to the directed variant by replacing each edge by the pair of arcs with opposite orientations. Still, it can be observed that the reduction in Proposition 10 strongly relies on the fact that the constructed graph G' has an articulation point u. Hence, it is natural to investigate the problem further imposing connectivity constraints on the input graphs. And indeed, it can be easily seen that Longest Path above Diameter is FPT on 2-connected undirected graphs.

▶ **Observation 11.** *Longest Path above Diameter can be solved in time $6.523^k \cdot n^{\mathcal{O}(1)}$ on undirected 2-connected graphs.*

Proof. Let (G,k) be an instance of Longest Path above Diameter where G is 2-connected. If $d = \text{diam}(G) \leq k$, we can solve the problem in time $2.554^{d+k} \cdot n^{\mathcal{O}(1)}$ by using the algorithm of Proposition 1 to check whether G has a path of length $d+k$. Note that $2.554^{d+k} \leq 2.554^{2k} \leq 6.523^k$. Otherwise, if $d > k$, consider a pair of vertices s and t with $\text{dist}_G(s,t) = d$. Because G is 2-connected, by Menger's theorem (see, e.g., [19]), G has a cycle C containing s and t. Since $\text{dist}_G(s,t) = d$ and $d \geq k+1$, the length of C is at least $d+k+1$. This implies that C contains a path of length $d+k$. ◀

However, the arguments from the proof of Observation 11 cannot be translated to directed graphs. In particular, if a directed graph G is 2-strongly-connected, it does not mean that for every two vertices u and v, G has a cycle containing u and v. We show the following theorem providing a full dichotomy for the complexity of Longest Path above Diameter on 2-strongly-connected graphs.

▶ **Theorem 12.** *On 2-strongly-connected directed graphs, Longest Path above Diameter with $k \leq 4$ can be solved in polynomial time, while for $k \geq 5$ it is* NP*-complete.*

In what remains of this section, we give some intuition behind the proof of Theorem 12; the details can be found in the full version of the paper [21]. To show the positive part of the theorem, it is sufficient to consider graphs with diameter greater than some fixed constant, because in graphs with smaller diameter the problem can be solved in linear time. For graphs with a sufficiently large diameter, we show that a path of length diameter plus four always exists. To construct such a path, we take the diameter pair (s, t) and employ 2-strong-connectivity of the graph to find two disjoint (s, t)-paths and two disjoint (t, s)-paths in the graph. We then show that out of the several possible ways to comprise a path out of the parts of these four paths, at least one always obtains a path of desired length. The most non-trivial case of this construction involves constructing two paths of length five, one ending in a vertex u that is at distance three from s and the other starting in a vertex v from which we can reach t using three arcs. We then concatencate these two paths using a specific (u, v)-path inbetween. Since (s, t) is a diameter pair, the length of any (u, v)-path is at least diameter minus six, so the length of the concatenation is at least diameter plus four. The other cases are analyzed in a similar fashion.

For the lower bound part of Theorem 12, the general idea of the proof is similar to that of Proposition 10. We aim to take a path-like gadget graph, then take a sufficiently large Hamiltonian Path instance and connect it to the middle of the gadget. However, while in the general case it suffices to simply take a path graph (Proposition 10), the 2-strongly-connected case is much more technically involved. First, we need a family of gadget graphs that are 2-strongly-connected, have arbitrarily large diameter, but each graph in the family does not have a path longer than diameter plus four. This, in fact, is exactly a counterexample to the positive part of Theorem 12, as the existence of such family of graphs proves that there cannot always be a path of length diameter plus four in a sufficiently large 2-connected directed graph. Additionally, for the reduction we need that graphs in this family behave like paths, specifically that the length of the longest path that ends in the "middle" of the gadget is roughly half of the diameter. Constructing this graph family is a main technical challenge of the theorem. After constructing the gadget graph family the proof is reasonably simple, as we take a 2-connected Hamiltonian Path instance, and connect it to the "middle" of a sufficiently large gadget graph. The connection is done by a simple 4-vertex connector gadget that ensures that the resulting graph is 2-strongly-connected, but only allows for paths that alternate at most once between the gadget graph and the starting instance.

5 Conclusion

We proved that if $\mathcal{C}$ is a class of directed graph such that p-Disjoint Paths is in P on $\mathcal{C}$ for $p = 3$, then Longest Detour is FPT on $\mathcal{C}$. However p-Disjoint Paths is NP-complete on directed graphs for every fixed $p \geq 2$ [29]. This leaves open the question of Bezáková et al. [7] about parameterized complexity of Longest Detour on general directed graphs. Even the complexity (P versus NP) of deciding whether a directed graph contains an (s, t)-path longer than $\mathrm{dist}_G(s, t)$ (the case of $k = 1$) remains open. Notice that Longest Detour is not equivalent to p-Disjoint Paths for $p = 3$ and, therefore, the hardness of p-Disjoint Paths does not imply hardness of Longest Detour.

Our result implies, in particular, that Longest Detour is FPT on planar directed graphs. There are various classes of directed graphs on which p-Disjoint Paths is tractable for fixed p (see, e.g., the book of Bang-Jensen and Gutin [3]). For example, by Chudnovsky, Scott, and Seymour [14], p-Disjoint Paths can be solved in polynomial time for every fixed

p on semi-complete directed graphs. Together with Theorem 4, it implies that LONGEST DETOUR is FPT on semi-complete directed graphs and tournaments. However, from what we know, these results could be too weak in the following sense. Using the structural results of Thomassen [49], Bang-Jensen, Manoussakis, and Thomassen in [4] gave a polynomial-time algorithm to decide whether a semi-complete directed graph has a Hamiltonian (s, t)-path for two given vertices s and t. Thus the real question is whether LONGEST DETOUR is in P on semi-complete directed graphs or tournaments.

The second part of our results is devoted to LONGEST PATH ABOVE DIAMETER. We proved that this problem is NP-complete for general graphs for $k = 1$ and showed that it is in FPT when the input graph is undirected and 2-connected. We established the complexity dichotomy for LONGEST PATH ABOVE DIAMETER for the case of 2-strongly-connected directed graphs by showing that the problem can be solved in polynomial time for $k \leq 4$ and is NP-complete for $k \geq 5$. This naturally leaves an open question for larger values of strong connectivity. The computational complexity of LONGEST PATH ABOVE DIAMETER on t-strongly connected graphs for $t \geq 3$ is open. For a very concrete question, is there a polynomial algorithm for LONGEST PATH ABOVE DIAMETER with $k = 5$ on graphs of strong connectivity 3?

References

1 Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 511–517. SIAM, 2010.

2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

3 Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs - Theory, Algorithms and Applications, Second Edition*. Springer Monographs in Mathematics. Springer, 2009.

4 Jørgen Bang-Jensen, Yannis Manoussakis, and Carsten Thomassen. A polynomial algorithm for hamiltonian-connectedness in semicomplete digraphs. *J. Algorithms*, 13(1):114–127, 1992. doi:10.1016/0196-6774(92)90008-Z.

5 Eli Berger, Paul Seymour, and Sophie Spirkl. Finding an induced path that is not a shortest path, 2020. arXiv:2005.12861.

6 Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding detours is fixed-parameter tractable. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 80 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.54.

7 Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding detours is fixed-parameter tractable. *SIAM J. Discret. Math.*, 33(4):2326–2345, 2019. doi:10.1137/17M1148566.

8 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *Journal of Computer and System Sciences*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

9 Hans L. Bodlaender. On linear time minor tests with depth-first search. *Journal of Algorithms*, 14(1):1–23, 1993. doi:10.1006/jagm.1993.1001.

10 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

11 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.

12 Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009. doi:10.1137/080716475.

13 Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 298–307. SIAM, 2007.

14 Maria Chudnovsky, Alex Scott, and Paul D. Seymour. Excluding pairs of graphs. *J. Comb. Theory, Ser. B*, 106:15–29, 2014. doi:10.1016/j.jctb.2014.01.001.

15 Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 24 of *LIPIcs*, pages 43–54. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.43.

16 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 197–206. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.29.

18 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE, 2011.

19 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

20 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

21 Fedor V. Fomin, Petr A. Golovach, William Lochet, Danil Sagunov, Kirill Simonov, and Saket Saurabh. Detours in directed graphs. *CoRR*, abs/2201.03318, 2022. arXiv:2201.03318.

22 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going far from degeneracy. *SIAM J. Discret. Math.*, 34(3):1587–1601, 2020. doi:10.1137/19M1290577.

23 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Parameterization Above a Multiplicative Guarantee. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.39.

24 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Algorithmic extensions of Dirac's Theorem. *CoRR*, abs/2011.03619, 2020. arXiv:2011.03619.

25 Fedor V. Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013. doi:10.1145/2428556.2428575.

26 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

27 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative families of product families. *ACM Trans. Algorithms*, 13(3):36:1–36:29, 2017. doi:10.1145/3039243.

28 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Long directed (s, t)-path: FPT algorithm. *Inf. Process. Lett.*, 140:8–12, 2018. doi:10.1016/j.ipl.2018.04.018.

29 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

30 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

31 Shivam Garg and Geevarghese Philip. Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1152–1166. SIAM, 2016. doi:10.1137/1.9781611974331.ch80.

32 Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. Vertex cover problem parameterized above and below tight bounds. *Theory of Computing Systems*, 48(2):402–410, 2011. doi:10.1007/s00224-010-9262-y.

33 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. *Journal of Computer and System Sciences*, 78(1):151–163, 2012. doi:10.1016/j.jcss.2011.01.004.

34 Gregory Z. Gutin and Viresh Patel. Parameterized traveling salesman problem: Beating the average. *SIAM J. Discrete Math.*, 30(1):220–238, 2016.

35 Gregory Z. Gutin, Arash Rafiey, Stefan Szeider, and Anders Yeo. The linear arrangement problem parameterized above guaranteed value. *Theory Comput. Syst.*, 41(3):521–538, 2007. doi:10.1007/s00224-007-1330-6.

36 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

37 Bart M. P. Jansen, László Kozma, and Jesper Nederlof. Hamiltonicity below Dirac's condition. In *Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 11789 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2019.

38 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

39 Ken-ichi Kawarabayashi and Stephan Kreutzer. The directed grid theorem. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 655–664. ACM, 2015. doi:10.1145/2746539.2746586.

40 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 58–67. Springer, 2006. doi:10.1007/11917496_6.

41 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2008. doi:10.1007/978-3-540-70575-8_47.

42 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Communications of the ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

43 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.

44 Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335–354, 1999. doi:10.1006/jagm.1998.0996.

45 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *Journal of Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/j.jcss.2008.08.004.

46 Burkhard Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. doi:10.1016/S0304-0208(08)73110-4.

47 Neil Robertson and Paul D. Seymour. Graph minors. XIII. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

48 Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM J. Comput.*, 23(4):780–788, 1994. doi:10.1137/S0097539792224061.

49 Carsten Thomassen. Hamiltonian-connected tournaments. *J. Comb. Theory, Ser. B*, 28(2):142–163, 1980. doi:10.1016/0095-8956(80)90061-1.

50 Dekel Tsur. Faster deterministic parameterized algorithm for k-path. *Theor. Comput. Sci.*, 790:96–104, 2019. doi:10.1016/j.tcs.2019.04.024.

51 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

Delay-Robust Routes in Temporal Graphs

Eugen Füchsle
Faculty IV, Algorithmics and Computational Complexity, TU Berlin, Germany

Hendrik Molter
Department of Industrial Engineering and Management,
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Rolf Niedermeier
Faculty IV, Algorithmics and Computational Complexity, TU Berlin, Germany

Malte Renken
Faculty IV, Algorithmics and Computational Complexity, TU Berlin, Germany

Abstract

Most transportation networks are inherently temporal: Connections (e.g. flights, train runs) are only available at certain, scheduled times. When transporting passengers or commodities, this fact must be considered for the the planning of itineraries. This has already led to several well-studied algorithmic problems on temporal graphs. The difficulty of the described task is increased by the fact that connections are often unreliable – in particular, many modes of transportation suffer from occasional delays. If these delays cause subsequent connections to be missed, the consequences can be severe. Thus, it is a vital problem to design itineraries that are *robust* to (small) delays. We initiate the study of this problem from a parameterized complexity perspective by proving its NP-completeness as well as several hardness and tractability results for natural parameterizations.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability; Mathematics of computing → Discrete mathematics

Keywords and phrases algorithms and complexity, parameterized complexity, time-varying networks, temporal paths, journeys

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.30

Related Version *Full Version*: `https://arxiv.org/abs/2201.05390` [17]

Funding *Hendrik Molter*: Supported by the ISF, grant No. 1070/20.
Malte Renken: Supported by the DFG, project MATE (NI 369/17).

1 Introduction

Finding a path between two vertices in a graph is one of the most fundamental problems in graph algorithms. In the rise in popularity of *temporal graphs* as a mathematical model [19, 20, 25, 24, 5], computing so-called *temporal paths* is one of the most important algorithmic problems in this area. Herein, a temporal graph is a graph whose edges are present only at certain, known points in time. For our purposes, it is specified by a set V of vertices and a set E of time arcs, where each time arc $(v, w, t, \lambda) \in E$ consists of a *start vertex* v, an *end vertex* w, a *time label* t, and a traversal time λ; this means that there is a (direct) connection from v to w starting at time t and arriving at time $t+\lambda$. Temporal graphs are prime models for many real-world networks: Social graphs, communication networks, and transportation networks are usually not static but vary over time.

The added dimension of time causes many aspects of connectivity to behave quite differently from static (i.e., non-temporal) graphs. In particular, the flow of goods or information through a temporal network has to respect time. More formally, it follows a *temporal walk* (or *path*, if every vertex is visited at most once), i.e., a sequence of time arcs

$(v_i, w_i, t_i, \lambda_i)_{i=1}^{\ell}$ where $v_{i+1} = w_i$ and $t_{i+1} \geq t_i + \lambda_i$ for all $i < \ell$. While inheriting many properties of their static counterparts, temporal walks exhibit certain characteristics that add a new level of complexity to algorithmic problems centered around them. For example, temporal connectivity is not transitive: the existence of a temporal walk from vertex u to w and a temporal walk from v to w does not imply the existence of a temporal walk from u to w. Furthermore, the temporal setting allows for several natural notions of an "optimal" temporal path [3].

As the finding of (optimal) temporal paths and walks constitutes the perhaps most important building block for (algorithmic) analysis of temporal networks, it has already been studied intensively [28, 3]. However, the temporal setting allows to model further natural constraints on temporal walks and paths that do not have a counterpart in the static setting. For example, recently the study of the computational complexity of finding temporal walks and paths that are subject to some waiting time constraints has been initiated [6, 1].

In this work, we investigate another very natural yet still unstudied temporal path variant, namely so-called *delay-robust* temporal paths. Real-world networks are often not perfect: Scheduled connections may be canceled or delayed. This immediately brings up the natural issue of robustness. To the best of our knowledge, this issue has so far only been analyzed with respect to cancellations [2], but not with respect to delays. We propose a model for delay-robust temporal paths and analyze natural structural and computational problems occurring in this context. Our main problem of interest is to determine whether there is a delay-robust temporal path between two vertices in a temporal graph.

Delay-Robust Route
Input: A temporal graph $\mathcal{G} = (V, E)$, two vertices $s, z \in V$ and $x, \delta \in \mathbb{N}$.
Question: Is there an x-delay-robust route from s to z in $\mathcal{G}$?

It remains to say how *delay-robustness* is understood. Although different notions are conceivable, we consider a sequence of vertices (called a *route*) in a temporal graph to be *x-delay-robust*, if there is a temporal path visiting the vertices in this sequence even if up to x time arcs are delayed by at most δ. We give a formal definition in Section 2.

This definition is motivated by the fact that changing the vertices may be costly for a number of reasons: storage or transhipment facilities may need to be newly allocated; if the new route passes through different jurisdictions, then new authorizations and documents have to be acquired; insurance policies might not cover alternative routes; the chosen packaging might no longer be adequate (e.g. when switching from rail to air transportation); or personnel might need to be moved. All these and many more issues are of much less concern when the chosen route can be kept and only the *schedule* has to be changed.

Related Work. Apart from the already mentioned work on finding temporal walks and paths, there has been extensive research on many other connectivity-related problems on temporal graphs [4, 14, 23]. Delays in temporal graphs have been considered as a modification operation to manipulate reachability sets [8, 26]. The individual delay operation considered in the mentioned work delays a single time arc and is similar to our notion of delays. The deletion of time arcs [26, 12, 11], the deletion of vertices [29, 16, 22], as well as reordering of time arcs [13] have also been considered as temporal graph modification operations to manipulate the connectivity properties of the temporal graph. The corresponding computational problems in all mentioned work are NP-hard and can be also considered as computing "robustness measures" for the connectivity in temporal graphs.

In companion work [18] we investigate the related problem where we ask whether two vertices remain connected even if up to x time arcs are delayed. Note that in this setting, the specific temporal path connecting the two vertices can visit different vertices for different delays. We show that this problem can be solved in polynomial time. We further investigate the problem variant where the delays occur dynamically during the "journey" from the start to the destination vertex. In this case the problem becomes PSPACE-complete if every vertex can be visited at most once and stays polynomial-time solvable, otherwise.

Our Contribution. We introduce the computational problem of finding routes that are robust under delays. We investigate its computational complexity with a focus on parameterized algorithms and hardness [10, 7].

We first give some structural results in Section 3, including that Delay-Robust Path is solvable in polynomial time if the underlying graph[1] is a forest. In Section 4, we show that Delay-Robust Path is NP-hard even if the underlying graph has constant bandwidth, which implies that it also has constant treewidth. We further show that Delay-Robust Path is W[1]-hard when parameterized by the combination of the feedback vertex number of the underlying graph and the number of delays. In Section 5, we present our general algorithmic results where we explore how the polynomial-time algorithm for underlying forests can be generalized. We give a polynomial-time algorithm for the case where we have a constant number of delays. We further give two FPT algorithms: one for the underlying feedback edge set number as a parameter and one for the combination of the so-called *timed* feedback vertex number [6] and the number of delays as a parameter.

Due to the lack of space, several proofs (marked with ($\star$)) had to be deferred to the full version [17].

2 Preliminaries

We abbreviate $\{1, 2, \dots, n\}$ as $[n]$ and $\{n, n+1, \dots, m\}$ as $[n, m]$. For any time arc $e = (v, w, t, \lambda_e)$, we denote the starting and ending vertices as $\mathrm{start}(e) = v$ an $\mathrm{end}(e) = w$, the time label as $\mathrm{t}(e) = t$, and the traversal time as $\lambda(e) = \lambda_e$. Furthermore, for any vertex v, τ_v^+ denotes the set of time steps where v has outgoing time arcs, and τ_v^- denotes the time steps with incoming time arcs. We set $\tau_v := \tau_v^+ \cup \tau_v^-$.

Given a temporal graph $\mathcal{G}$, we denote by T the maximum time label of all time arcs in $\mathcal{G}$. When removing all time information and directions from the time arcs of a temporal graph $\mathcal{G} = (V, E)$, the resulting (static & undirected) graph $G_{\mathrm{u}}(\mathcal{G}) = (V, E')$ with $E' = \{\{v, w\} \mid (v, w, t, \lambda) \in E\}$ is called the *underlying graph* of $\mathcal{G}$.

Delays. We distinguish two different types of delays. Both are applied to a single time arc e and delay it by a natural number δ. A *starting delay* increases the time label $\mathrm{t}(e)$ by δ while a *traversal delay* increases the traversal time $\lambda(e)$ by δ. In the example of a railway network, a starting delay would correspond to a delayed departure at a station whereas a traversal delay would describe a delay occurring on the way between two stations.

For a given set $D \subseteq E$ of *delayed arcs*, a sequence of time arcs $(v_i, w_i, t_i, \lambda_i)_{i=1}^{\ell}$ is called a D-starting-delayed temporal walk resp. a D-traversal-delayed temporal walk if it is a temporal walk in the temporal graph obtained from $\mathcal{G}$ by applying starting delays resp. traversal delays to all time arcs in D. (We omit D as well as the type of delay when they are clear from context.) Note that a traversal-delayed temporal walk is always also a temporal walk in $\mathcal{G}$, which is not necessarily true for a starting-delayed temporal walk.

[1] The *underlying graph* of a temporal graph is the undirected static graph obtained by connecting all vertices that are connected by a time arc.

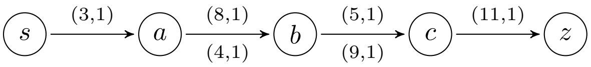
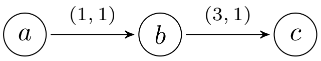


Figure 1 An example temporal graph where (s, a, b, c, z) is a traversal- and starting-delay-robust temporal route for $x = 1$ and $\delta = 3$. No matter which time arc is delayed, there is a temporal path through these vertices in that order. On the other hand, the route ceases to be delay-robust for $\delta \geq 5$, as delaying the first arc demonstrates.

As an example consider the following temporal walk with edges labeled by $(\mathrm{t}(e), \lambda(e))$:



When delaying the first time arc by 1, i.e. when setting $\delta = 1$ and $D = \{(a, b, 1, 1)\}$, then this is also a starting-delayed as well as a traversal-delayed temporal walk: Due to the delay, the first time arc arrives in b at time step $2 + \delta = 3$ which is no later than the departure of the second time arc. However, if we instead set $\delta = 2$ and $D = \{(a, b, 1, 1), (b, c, 3, 1)\}$, then it is still a starting-delayed temporal walk but no longer a traversal-delayed temporal walk, because the first time arc only reaches b at time 4.

We say a sequence R of vertices forms a *(delayed) route* if there is a (delayed) temporal walk which *follows* R, that is, which visits exactly the vertices of R in the given order. Generally, a temporal walk or route from vertex s to vertex z is also called a *temporal* (s, z)*-walk* or (s, z)*-route*. A *(delayed) temporal path* is a (delayed) temporal walk where no vertex is visited twice.

Robustness. We say that a temporal route is *traversal-delay-robust* resp. *starting-delay-robust* for a given number x of delays if it is a D-traversal-delayed resp. D-starting-delayed temporal route for all delay sets D of size $|D| \leq x$. Of course, this also depends on the value of δ. An example can be seen in Figure 1.

We now have all the ingredients for the formal definition of our main problem, Delay-Robust Route, as given in Section 1. In this definition, we did not specify whether traversal- or starting-delay is used. The reason for that is that we will show in Section 3 that the distinction is meaningless because both problem variants are equivalent. In the meantime, however, we will refer to them as TD-Delay-Robust Route and SD-Delay-Robust Route.

3 Structural Results and Recognizing Robust Routes

In this section, we derive some important properties of delay-robust routes.

3.1 Structural Results

We begin by investigating the distinction between walks and paths. Clearly, from any temporal walk one can obtain a temporal path by eliminating all circular subwalks. This leads to the following lemma, which holds for traversal as well as starting delays, and for all delay sizes x and delay times δ and will come in handy later.

▶ **Lemma 1.** *Let s and z be two vertices. If there is a delay-robust (s, z)-route, then there is a delay-robust (s, z)-route without repeated vertices.*

Proof. If there is a delay-robust (s,z)-route $R=(v_i)_{i=1}^k$, then for each delay of size at most x there is a delayed temporal walk traversing $v_1,v_2,\dots,v_k$ in that order. Each of these delayed temporal walks can be turned into a delayed temporal walk by eliminating circular subwalks. All the delayed temporal paths obtained in this way follow the same sequence of vertices, making this sequence a delay-robust (s,z)-route without repeated vertices. ◀

By virtue of Lemma 1, we will subsequently assume routes to not contain repeated vertices.

Next, we turn towards proving the equivalence of SD-Delay-Robust Route and TD-Delay-Robust Route. We start with some important observations. The first one is that every traversal-delayed temporal walk is also a starting-delayed temporal walk.

▶ **Lemma 2.** *Let P be a traversal-delayed temporal walk for some delay set D and $\delta\in\mathbb{N}$. Then P is also a starting-delayed temporal walk for D and δ.*

Proof. Let $P=(v_i,w_i,t_i,\lambda_i)_{i=1}^{\ell}$. This means that

$$t_i+\lambda_i+[e_i\in D]\cdot\delta\le t_{i+1}$$

for all $i\le\ell-1$, where $[e_i\in D]=\begin{cases}1 & \text{if } e_i\in D\\ 0 & \text{otherwise}\end{cases}$ denotes the Iverson bracket. Thus

$$t_i+\lambda_i+[e_i\in D]\cdot\delta\le t_{i+1}+[e_{i+1}\in D]\cdot\delta$$

which shows that P is a starting-delayed temporal walk. ◀

While the converse of Lemma 2 is generally not true, the following weaker statement holds.

▶ **Lemma 3.** *Let R be a route, $\delta\in\mathbb{N}$ and D a minimal delay set such that R is not a D-traversal-delayed route. Then R is not a D-starting-delayed route either.*

Proof. Suppose for contradiction that R was a D-starting-delayed route. Then there is a D-starting-delayed temporal walk $P=(e_i)_{i=1}^{\ell}=(v_i,w_i,t_i,\lambda_i)_{i=1}^{\ell}$ that follows R, i.e.,

$$t_i+\lambda_i+[e_i\in D]\cdot\delta\le t_{i+1}+[e_{i+1}\in D]\cdot\delta$$

for all $i\le\ell-1$. Since R is not a traversal-delayed route, P is not a traversal-delayed temporal path. Thus there exists an index $j\le\ell-1$ with

$$t_j+\lambda_j+[e_j\in D]\cdot\delta>t_{j+1}$$

and we may assume j to be chosen maximally. This implies that

$$t_{j+1}<t_j+\lambda_j+[e_j\in D]\cdot\delta\le t_{j+1}+[e_{j+1}\in D]\cdot\delta,$$

which in turn implies that $e_{j+1}\in D$. By maximality of j, $P'=(e_i)_{i=j+1}^{\ell}$ is a traversal-delayed temporal path. Thus, for any traversal-delayed temporal path $Q=(v_i,w_i,t'_i,\lambda'_i)_{i=1}^{j}$ following $(v_i)_{i=1}^{j+1}$, we must have $t'_j+\lambda'_j+[(v_j,w_j,t'_j,\lambda'_j)\in D]\cdot\delta>t_{j+1}$, for otherwise its concatenation with P' would contradict the fact that R is not a traversal-delayed route. Therefore, R is also not a D'-traversal-delayed temporal vertex walk, where $D'=D\setminus\{e_{j+1}\}$. This contradicts the minimality of D. ◀

Using Lemmas 2 and 3, we can now prove the following.

▶ **Theorem 4.** *TD-Delay-Robust Route = SD-Delay-Robust Route.*

Proof. Let $\mathcal{G} = (V, E)$ be a temporal graph, $s, z \in V$ be a start and an end vertex, and $x, \delta \in \mathbb{N}$. If $(\mathcal{G}, s, z, x, \delta)$ is a no-instance of SD-Delay-Robust Route, then for every (s, z)-route R, there exists a set D of $|D| \leq x$ time arcs such that there is no D-starting-delayed temporal path following R. By Lemma 2, there is then also no D-traversal-delayed temporal path following R, thus $(\mathcal{G}, s, z, x, \delta)$ is a no-instance of TD-Delay-Robust Route.

Conversely, if $(\mathcal{G}, s, z, x, \delta)$ is a no-instance of TD-Delay-Robust Route, then for every (s, z)-route R there exists a set D of $|D| \leq x$ time arcs such that R is no D-traversal-delayed route. We may assume D to be minimal to that respect. Then Lemma 3 gives us that R is no D-starting-delayed route, making $(\mathcal{G}, s, z, x, \delta)$ a no-instance of SD-Delay-Robust Route. ◀

Theorem 4 allows us now to drop the distinction between the two delay types and speak simply of Delay-Robust Route. For the remainder of this paper we will mostly work with traversal delays, for they are slightly easier to handle.

3.2 Recognizing Robust Routes

Since a route can be followed by an exponential number of different temporal walks, it is not immediately clear whether delay-robustness can be efficiently checked. The following theorem says that this is the case, and Delay-Robust Route is thus contained in NP.

▶ **Theorem 5** ($\star$)**.** *For any given $x, \delta \in \mathbb{N}$, one can determine in $\mathcal{O}(nmx^2 + m \log m)$ time whether a given route R in a temporal graph $\mathcal{G}$ is x-delay-robust, where n is the number of vertices of R and m is the number of time arcs connecting consecutive vertices of R.*

Theorem 5 also gives us a polynomial-time algorithm to solve Delay-Robust Route on temporal graphs with underlying forest: As any vertex pair (s, z) is connected by at most one route, we only need to test the delay robustness of that route.

In the remainder of this section, we will prove Theorem 5. The basic idea is that a route is delay-robust for a worst-case delay if and only if it is delay-robust for all delays. This worst-case delay can be computed in polynomial time using a dynamic program.

First, we introduce the term *earliest arrival time* for a given route. A route $R = (v_i)_{i=1}^k$ requires that there is at least one temporal path following R. The *earliest arrival time* is then the arrival time of the temporal path that arrives earliest. Formally, we define the earliest arrival time as follows. Let $\mathcal{P} = \{P_i\}_{i=1}^\ell$ be the set of temporal paths following R with $P_i = (e_1^{(i)}, e_2^{(i)}, \ldots, e_{k-1}^{(i)})$. The *earliest arrival time* of R then is defined as the earliest arrival time of any temporal path in $\mathcal{P}$, i.e., as $\min \left\{ t(e_{k-1}^{(i)}) + \lambda(e_{k-1}^{(i)}) \;\middle|\; i \leq \ell \right\}$. Analogously, if $R = (v_i)_{i=1}^k$ is a delayed route for the delay set $D \subseteq E$ and delay time $\delta \in \mathbb{N}$, and if $\mathcal{P}$ as above is the set of delayed temporal paths following R, then the *earliest delayed arrival time* of R is $\min \left\{ t(e_{k-1}^{(i)}) + \lambda(e_{k-1}^{(i)}) + [e_{k-1}^{(i)} \in D] \cdot \delta \;\middle|\; i \leq \ell \right\}$.

We then define the *worst-case arrival time* of a route $R = (v_i)_{i=1}^k$ for a given delay size x and delay time δ as the maximum earliest delayed arrival time of R, taken over all delay sets D with $|D| \leq x$. (If R is not x-delay-robust, then we define the worst-case arrival time to be ∞.)

Now that we defined the worst-case arrival time, we show how to compute it. Let $R_j = (v_i)_{i=1}^j$ denote the prefix routes of R. The dynamic program computes table entries $A_{R_j}[y]$ iteratively for all $j \leq k$ and $y \leq x$, where $A_{R_j}[y]$ stores the worst-case arrival time of R_j for y delays.

We begin with the single-vertex route $R_1 = (v_1)$, setting $A_{R_1}[y] = 0$ for all y since the empty temporal path is always available to go from v_1 to v_1. Our goal is then to inductively compute A_{R_j} from $A_{R_{j-1}}$.

Consider the situation where we want to get from v to w in a single step, starting at time t. Then the set of available time arcs is $E(v, w, t) = \{(v, w, t', \lambda) \in E \mid t' \geq t\}$. Suppose $E(v, w, t) = \{a_i\}_{i=1}^{\ell}$ where $\mathrm{t}(a_i) + \lambda(a_i) \leq \mathrm{t}(a_{i+1}) + \lambda(a_{i+1})$ for all i. Now if up to y delays occur, then the latest time at which we will reach w is

$$\alpha(v, w, t, y) := \min\{\mathrm{t}(a_1) + \lambda(a_1) + \delta, \mathrm{t}(a_{y+1}) + \lambda(a_{y+1})\}.$$

Here, the worst case occurs if a_1 through a_y are all delayed.

Using this fact, we can now compute the table entries A_{R_i} from $A_{R_{i-1}}$ as follows.

$$A_{R_i}[y] = \max_{0 \leq y' \leq y} \{\alpha(v_{i-1}, v_i, A_{R_{i-1}}[y'], y - y')\}.$$

The idea here is that some number $y' \leq y$ of delays will occur between v_{i-1} and v_i, while the other $y - y'$ delays can occur somewhere along R_{i-1}.

The formal proof that $A_R[x]$ contains the solution to the Delay-Robust Route instance and that it can be computed in the specified time is deferred to the full version [17].

4 A Reduction Framework for Delay-Robust Route

In this section, we investigate the computational hardness of Delay-Robust Route with a particular attention to parameterized hardness with respect to "distance to forest" parameters. The goal is to lay out the ground for potential generalization of the algorithm presented in Section 3.2. We introduce a new problem Multi-Colored Monotone SAT in Section 4.1 and design a polynomial-time reduction to Delay-Robust Route. We will use this as an intermediate problem for reductions from 3-SAT and Multi-Colored Clique in Section 4.2 to show NP-hardness and parameterized hardness results.

4.1 Multi-Colored Monotone SAT

The problem Multi-Colored Monotone SAT is a Satisfiability variant where the variables are partitioned into "color classes" and only one variable from each color may be set to true. Furthermore, we do not make any assumptions on the Boolean formula other than that all variables appear non-negated. Formally, the we define the problem as follows.

Multi-Colored Monotone SAT (MCMSAT)

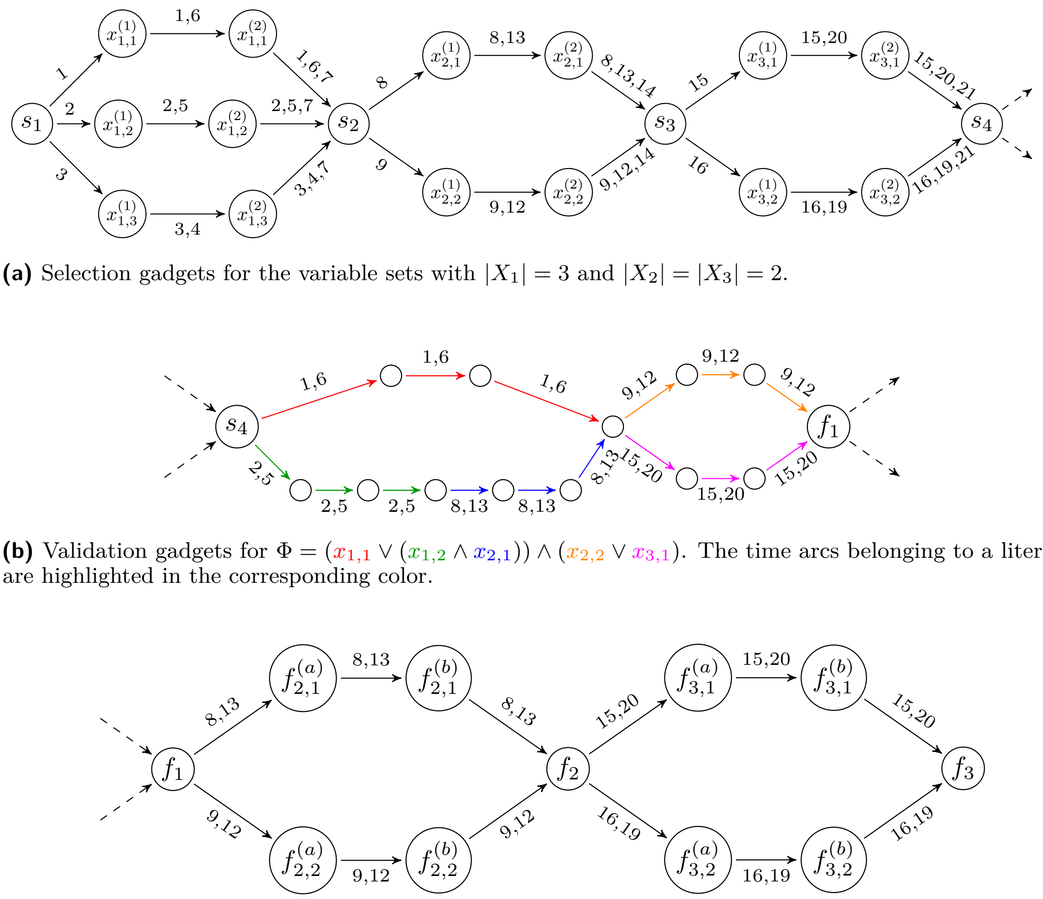
Input: Disjoint sets of variables $X_1, X_2, \ldots, X_n$ and a boolean formula Φ only consisting of positive literals and the operators $\wedge$ and $\vee$.

Question: Is there a satisfying truth assignment for Φ where exactly one variable from each X_i for $i \in [n]$ is true?

We have the following theorem.

▶ **Theorem 6** ($\star$). *MCMSAT $\leq_{\mathrm{m}}^{\mathrm{poly}}$ Delay-Robust Route.*

We describe the reduction behind Theorem 6 here, but defer most of the formal correctness proofs to the full version [17].



(a) Selection gadgets for the variable sets with $|X_1| = 3$ and $|X_2| = |X_3| = 2$.

(b) Validation gadgets for $\Phi = (x_{1,1} \vee (x_{1,2} \wedge x_{2,1})) \wedge (x_{2,2} \vee x_{3,1})$. The time arcs belonging to a literal are highlighted in the corresponding color.

(c) Finalization gadgets for $n = 3$.

Figure 2 An example temporal graph resulting from a Multi-Colored Monotone SAT reduction. Dummy time arcs are omitted. The instance has $n = 3$ disjoint variable sets.

Let $I = ((X_1, X_2, \dots, X_n), \Phi)$ be an instance of MCMSAT. We will construct a temporal graph $\mathcal{G} = (V, E)$ and an instance $I' = (\mathcal{G}, s, z, \delta, x)$ of Delay-Robust Route so that I is a yes-instance of MCMSAT if and only if I' is a yes-instance of Delay-Robust Route. All time arcs in $\mathcal{G}$ have a traversal time of 0. Thus, we abbreviate time arcs as 3-tuples (v, w, t). In figures we omit the traversal time and label arcs only with their time step.

Let $X_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,|X_i|}\}$ for all $i \in [n]$. Furthermore, let $m := \max_i |X_i|$ denote the largest cardinality of a variable set X_i. The temporal graph consists of chained *selection gadgets* for each variable set X_i, a recursively constructed *validation gadget* and chained *finalization gadgets* for each variable set X_i. The selection gadgets are used to select the variable from X_i that is assigned to true, for each $i \in [n]$. Then the validation gadgets check whether the formula is satisfied under the selected truth assignment. If this is not the case, then a connection breaks at latest in the finalization gadgets and the target vertex can not be reached. The gadgets use an offset $o_i := (2m + 1) \cdot (i - 1)$. We set the delay time to $\delta = 1$ and the number of delays to $x = 2 \cdot n - 1$. Figure 2 shows examples for all gadget types.

Selection Gadgets. The selection gadgets are used to select one variable $x_{i,a}$ from each set X_i. For each set X_i, we add a vertex s_i to V and one additional vertex s_{n+1}. For each set X_i and each variable $x_{i,a} \in X_i$, we add the vertices $x^{(1)}_{i,a}$ and $x^{(2)}_{i,a}$ to V. Moreover, we add the following time arcs to E so that there is one route from s_i to s_{i+1} for this variable $x_{i,a}$:

$$s_i \xrightarrow{o_i+a} x^{(1)}_{i,a} \xrightarrow{o_i+a,o_{i+1}-a} x^{(2)}_{i,a} \xrightarrow{o_i+a,o_{i+1}-a,o_{i+1}} s_{i+1}$$

Taking this sub-route corresponds to setting the variable $x_{i,a}$ to true. Additionally, for each of the three underlying arcs we add a *dummy time arc* for each time step $t \in [o_{i+1} - 1]$. If the sub-route $s_i \to x^{(1)}_{i,a} \to x^{(2)}_{i,a} \to s_{i+1}$ is chosen, then the worst-case arrival time from s_1 to s_{i+1} is $o_i + a$ for $2 \cdot (i-1)$ delays and $o_{i+1} - a = o_i + 2 \cdot m + 1 - a$ for $2 \cdot (i-1) + 1$ delays. Any sub-route is a Pareto optimum: while one arrives earlier for $2 \cdot (i-1)$ delays, another arrives earlier for $2 \cdot (i-1) + 1$ delays. An example for chained selection gadgets can be seen in Figure 2a.

Validation Gadgets. The validation gadgets are used to check whether the formula Φ is satisfied under the selected truth assignment. We will add a fresh vertex f_1 to V which is the start of the validation gadgets. The validation gadget for Φ will be constructed with s_{n+1} as a start vertex and f_1 as an end vertex. Given a start vertex v and an end vertex w, we can recursively construct the validation gadget for a formula Φ in the following way:

1. $\Phi = x_{i,a}$ is a single positive literal.
   We add two fresh vertices $\ell^{(1)}_{i,a}$ and $\ell^{(2)}_{i,a}$ to V. We add the following time arcs, so that there is a connection from v to w:

   $$v \xrightarrow{o_i+a,o_{i+1}-a} \ell^{(1)}_{i,a} \xrightarrow{o_i+a,o_{i+1}-a} \ell^{(2)}_{i,a} \xrightarrow{o_i+a,o_{i+1}-a} w$$

   Additionally for all three underlying arcs we add a *dummy time arc* for each time step $t \in [o_{n+1} - 1] \setminus [o_i, o_{i+1} - 1]$. We call this constructed part of the validation gadget a *literal gadget*. If the variable $x_{i,a}$ has been selected in the selection gadgets, then traversing this literal gadget does not affect the worst-case arrival time with respect to the number of delays. However, if $x_{i,a}$ has not been selected there is a delay that breaks the connection at latest in the finalization gadgets.
2. $\Phi = \Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_k$ is a conjunction of k sub-formulae.
   We add a fresh vertex c_i to V for all $i \in [k-1]$. Then the validation gadgets for all sub-formulae Φ_i are constructed, with c_{i-1} as the start and c_i as the end vertex, where $c_0 = v$ and $c_k = w$. Thus, the gadgets for the sub-formulae are connected in a row, and to traverse the temporal graph from v to w all gadgets for the sub-formulae have to be traversed.
3. $\Phi = \Phi_1 \vee \Phi_2 \vee \ldots \vee \Phi_k$ is a disjunction of k sub-formulae.
   We construct the validation gadgets for all sub-formulae Φ_i with v as the start and w as the end vertex. Thus, the gadgets for the sub-formulae are connected in parallel, and to traverse the temporal graph from v to w one gadget for a sub-formulae has to be traversed.

An example for a valid gadget can be seen in Figure 2b.

Finalization Gadgets. The finalization gadgets are similar to the selection gadgets for all sets X_2 to X_n. For each variable set X_i for $i \in [2, n]$ we add a vertex f_i to V. For each variable $x_{i,a} \in X_i$ we add the vertices $f^{(1)}_{i,a}$ and $f^{(2)}_{i,a}$ to V and add the following time arcs:

$$f_{i-1} \xrightarrow{o_i+a,o_{i+1}-a} f^{(1)}_{i,a} \xrightarrow{o_i+a,o_{i+1}-a} f^{(2)}_{i,a} \xrightarrow{o_i+a,o_{i+1}-a} f_i$$

Again for all three underlying arcs and each time step $t \in [o_{n+1} - 1] \setminus [o_i, o_{i+1} - 1]$ we add a dummy time arc. An example for finalization gadgets can be seen in Figure 2c.

The start and end vertices for our Delay-Robust Route-instance are s_1 and f_n, respectively.

We defer the proof that the constructed Delay-Robust Route instance is equivalent to the given Multi-Colored Monotone SAT instance to the full version [17].

4.2 Applications of the Framework

Next, we use our previous result that MCMSAT $\leq_{\mathrm{m}}^{\mathrm{poly}}$ Delay-Robust Route (Theorem 6) to show that Delay-Robust Route is NP-complete even if the underlying graph has bandwidth 3. The *bandwidth* $\mathrm{bw}(G)$ of a graph G is the smallest number b such that the vertices of G can be placed at distinct integer points along a line so that the length of the longest edge is b. The bandwidth of a graph upper-bounds both the graph's pathwidth and treewidth [27]. Formally, we show the following result by using an appropriate polynomial-time reduction from the NP-complete 3-SAT problem [21] to MCMSAT.

▶ **Theorem 7** ($\star$). *Delay-Robust Route is NP-complete for all fixed $\delta \geq 1$, maximum traversal times $\lambda_{\max} \geq 0$, and bandwidths of the underlying graph $\mathrm{bw}(G_{\mathrm{u}}(\mathcal{G})) \geq 3$.*

Next, we show W[1]-hardness of Delay-Robust Route for the feedback vertex set of the underlying graph, the length of a delay-robust temporal path, and the number of delays combined. To this end, we give a parameterized polynomial-time reduction from Multi-Colored Clique [15] to Delay-Robust Route. Again we use MCMSAT as an intermediate problem and use Theorem 6. Formally, we show the following result.

▶ **Theorem 8** ($\star$). *Delay-Robust Route is* W[1]*-hard with respect to $x + L + f$ where x is the number of delays, L is the length of a longest s-z path in $G_{\mathrm{u}}(\mathcal{G})$, and f is the feedback vertex number of $G_{\mathrm{u}}(\mathcal{G})$.*

The presented hardness results show that we presumably cannot generalize Theorem 5 to an FPT result for parameters such as the treewidth of the underlying graph or the feedback vertex number of the underlying graph.

5 Parameterized Algorithms

In Section 4, we presented several hardness results. Here, we present our algorithmic results for general input graphs which can be seen as different ways to generalize Theorem 5. We start with an XP-algorithm for the number of delays as a parameter and then present two FPT algorithms for "distance to forest" parameters.

5.1 Number of Delays

In what follows, we present an algorithm similar to Dijkstra's algorithm [9]. Starting at the source vertex s, it finds all optimal temporal (s, v)-routes by expanding each optimum by one step per iteration. However, as we have seen in the polynomial-time reductions in Section 4, there can be many (s, z)-routes that are Pareto-optimal with respect to the arrival time for a given number of delays. We use the dynamic program from Theorem 5 to extend the paths by a single time arc. Our main result of this section is that Delay-Robust Route admits an XP-algorithm with respect to the number x of delays. Theorem 8 implies that we presumably cannot improve this to an FPT result for this parameter. Formally, we show the following (in the remainder of this subsection, we provide a sketch of proof).

▶ **Theorem 9** ($\star$). *Delay-Robust Route can be solved in $\mathcal{O}(|V|^3 \cdot |E|^{2x} \cdot x^2)$ time, where x is the number of allowed delays.*

For each route, its *arrival time vector* $\vec{t} = (t_0, t_1, \dots, t_x)$ is a vector of $x+1$ time steps where t_y is the worst-case arrival time for y delays. We define a partial order $\preceq$ to compare arrival time vectors. For $\vec{t} = (t_0, t_1, \dots, t_x)$ and $\vec{t'} = (t'_0, t'_1, \dots, t'_x)$, set $\vec{t} \preceq \vec{t'}$ if and only if $t_y \leq t'_y$ for all $y \leq x$. This partial order can be used to decrease the set of prefix paths that need to be considered due to the following observation.

▶ **Observation 10.** *Let $\mathcal{G} = (V, E)$ be a temporal graph, let $s, v, z \in V$ be three vertices, and P_1 and P_2 be two delay-robust (s,v)-routes with the arrival time vectors $\vec{t_1} \preceq \vec{t_2}$. If there is a delay-robust (s,z)-route P so that $P = P_2 \circ P'$, then $P_1 \circ P'$ is also a delay-robust route. Additionally, if there is a delay-robust (s,v)-route, then there is one whose arrival time vector is minimal among all (s,v)-routes.*

Since for any delay one can arrive earlier in vertex v by using the route P_1 compared to P_2, replacing the prefix P_2 by P_1 still guarantees delay-robustness.

We define a table A with entries for every vertex of $\mathcal{G}$. The table entry $A[v]$ contains a set of arrival time vectors for (s,v)-routes. We will only store vectors that are minimal with respect to $\preceq$, since we do not need to consider others due to Observation 10. Thus, the set $A[v]$ will represent the Pareto front of routes from s to v.

Furthermore, we define a priority queue Q that contains tuples $(v, \vec{t})$ of vertices and arrival time vectors. The queue is sorted according by the arrival time vectors according to $\preceq$. The queue elements $(v, \vec{t})$ contain the prefix routes from where a search should be expanded.

We initialize the table A as follows:

$$A[v] = \begin{cases} \{(0, \dots, 0)\}, & \text{if } v = s \\ \emptyset, & \text{otherwise.} \end{cases}$$

The start vertex s can always be reached through the empty path. For all other vertices there is initially no route stored. Furthermore, we initialize the queue Q with the tuple $(s, (0, \dots, 0))$.

To compute the table entries we repeatedly pop the first element $(v, \vec{t})$ from Q and propagate possible delay-robust routes from there. If $(v, \vec{t})$ is in the queue, then this means that there is a delay-robust (s,v)-route P with the arrival time vector $\vec{t}$.

Let $\text{next}_v := \{w \mid (v, w, t, \lambda) \in V\}$ denote the set of vertices reachable from v by a single time arc. For all $w \in \text{next}_v$, we compute the arrival time vector $\vec{t'} = (t'_0, t'_1, \dots, t'_x)$ of $P' = P \circ (w)$ using the dynamic program described in Section 3.2: The arrival time vector of P' is simply the table row $A_{P'}$ and P' is y-delay-robust if and only if $A_{P'}[y] < \infty$.

As an optimization, we can round up the arrival time entries to the next time step in τ^+_w, i.e. replace t'_y by

$$\hat{t'_y} = \min_t \{t \in \tau^+_w \mid t \geq t'_y\}.$$

This rounding does not change the delay-robustness of any route since no temporal walk can leave w between time t'_y and $\hat{t'_y}$.

If P' is x-delay-robust, then we can add $\vec{t'}$ to the set $A[w]$, unless $A[w]$ already contains a smaller arrival time vector. We then delete all $\vec{t''}$ with $\vec{t'} \preceq \vec{t''}$ from $A[w]$ and also remove the corresponding elements $(w, \vec{t''})$ from the queue Q. Finally, we insert $(w, \vec{t'})$ into Q.

Once the queue Q is empty, we have investigated all x-robust prefix routes that might eventually lead to z. There is then a delay-robust (s, v)-route if and only if $A[z] \neq \emptyset$.

We defer the correctness proof for the presented algorithm as well as the running time analysis to the full version [17].

5.2 Timed Feedback Vertex Number

In this section, we explore another way to generalize Theorem 5. We present an FPT algorithm for the so-called *timed feedback vertex number* (introduced by Casteigts et al. [6]) and the number x of delays combined. Intuitively, the timed feedback vertex number is the minimum number of "vertex appearances" that need to be removed from the temporal graph to turn its underlying graph into a forest. Formally, it is defined as follows.

Let $\mathcal{G}$ be a temporal graph and $X \subseteq V \times [T]$ a set of *vertex appearances*. Then we write $\mathcal{G} - X := (V, E')$, where $E' = E \setminus \{(v, w, t, \lambda) \mid (v, t) \in X \vee (w, t) \in X\}$. A *timed feedback vertex set* of $\mathcal{G}$ is a set $X \subseteq V \times [T]$ of vertex appearances such that $G_{\mathrm{u}}(\mathcal{G} - X)$ is cycle-free. The *timed feedback vertex number* of a temporal graph $\mathcal{G}$ is the minimum cardinality of a timed feedback vertex set of $\mathcal{G}$.

▶ **Theorem 11** (⋆). *Delay-Robust Route can be solved in* $2^{\mathcal{O}(xf \log f)} \cdot (|V| + |E|)^{\mathcal{O}(1)}$ *time, where* f *is the timed feedback vertex number of the underlying graph.*

In the following, we give a description of the main steps of the algorithm we use to obtain the above result. The algorithm follows a simple "guess and check"-approach.

1. Compute a minimum timed feedback vertex set X of the input graph using an algorithm provided by Casteigts et al. [6].
2. Let $\hat{X} = \{v \mid (v, t) \in X\}$. Iterate over all partitions $\hat{X}_0 \uplus \hat{X}_1 \uplus \hat{X}_2 \uplus \hat{X}_3 = \hat{X}$ of $\hat{X}$. We distinguish two types of neighbors of a vertex. A neighbor connected by a time arc that is preserved in $\mathcal{G} - X$ is called a "forest neighbor", while other neighbors are called "feedback neighbors". Intuitively, in this step we guess for each vertex whether its predecessor resp. successor in the route is a feedback neighbor or a forest neighbor, leading to the following four cases:
   - The route does not contain v or the predecessor and successor of v in the route are forest neighbors of v (then $v \in \hat{X}_0$),
   - the predecessor of v in the route is a forest neighbor v, and the successor of v in the route is a feedback neighbor of v (then $v \in \hat{X}_1$),
   - the predecessor of v in the route is a feedback neighbor v, and the successor of v in the route is a forest neighbor of v (then $v \in \hat{X}_2$), or
   - the predecessor and successor of v in the route are feedback neighbors of v (then $v \in \hat{X}_3$).
3. Iterate over all orders on $\hat{X}_1 \cup \hat{X}_2 \cup \hat{X}_3$. Intuitively, in this step we guess in which order the vertices appear in the route.
4. Let $\hat{T} = \{t, t+\delta \mid \exists w \in V : (w, t) \in X\} \cup \{\infty\}$ be the *relevant* time steps. For each vertex $v \in \hat{X}_1 \cup \hat{X}_2 \cup \hat{X}_3$, iterate over all *delay profiles* $(t_1, t_2, \ldots, t_x) \in \hat{T}^x$. Intuitively, here we guess for each delay size i the smallest relevant time t_i which is at least the worst-case arrival time at v.
5. Use Theorem 5 to find route segments that respect the guessed delay profiles between consecutive vertices in $\hat{X}_1 \cup \hat{X}_2 \cup \hat{X}_3$ and which can be combined to an x-delay-robust (s, z)-route.

A detailed description of the last step and a sketch of the main ideas for the correctness proof and running time analysis are deferred to the full version [17].

5.3 Underlying Feedback Edge Number

In this section, we show that Delay-Robust Route admits an FPT-algorithm with respect to the feedback edge number of the underlying graph. Given a (static) undirected graph $G = (V, E)$, a feedback edge set $F \subseteq E$ is a set of edges, so that $G - F$ is acyclic. The *feedback edge number* is the cardinality of a minimum feedback edge set of G. Formally, we show the following.

▶ **Theorem 12** (⋆). *Delay-Robust Route can be solved in* $2^{\mathcal{O}(f)} \cdot (|V| \cdot |E| \cdot x^2) + \mathcal{O}(|E| \cdot \log |E|)$ *time, where* f *is the feedback edge number of the underlying graph.*

Casteigts et al. [6] designed an FPT-algorithm for the so-called Restless Temporal Path parameterized by the feedback edge number of the underlying graph. This algorithm can be applied to Delay-Robust Route as well with minor modifications in order to prove Theorem 12.

6 Conclusion

We modeled a naturally motivated path-finding problem taking into account delays by means of (algorithmic) temporal graph theory. For our central problem, Delay-Robust Route, we found computational hardness already for some tree-like underlying (static) graphs. While having provided a few encouraging parameterized tractability results, we leave plenty of room for further investigations into this direction. In particular, we left open what happens for the special case when the number of time labels per edge is bounded from above (in parameterized complexity terms, taking this as a parameter). Recall that our central hardness reduction needs many time labels. Moreover, the parameters vertex cover number or timed feedback vertex set number [6] (as a single parameter) deserve investigations as well. Rather from a modeling perspective, one might vary the basic problem by e.g. considering a *global* delay budget or other variations of the delay concept.

References

1 Matthias Bentert, Anne-Sophie Himmel, André Nichterlein, and Rolf Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020. doi:10.1007/s41109-020-00311-0.

2 Kenneth A Berman. Vulnerability of scheduled networks and a generalization of Menger's theorem. *Networks*, 28(3):125–134, 1996. doi:10.1002/(SICI)1097-0037(199610)28:3<125::AID-NET1>3.0.CO;2-P.

3 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003. doi:10.1142/S0129054103001728.

4 Sebastian Buß, Hendrik Molter, Rolf Niedermeier, and Maciej Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2020*, pages 2084–2092. ACM, 2020. doi:10.1145/3394486.3403259.

5 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.

6 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021. doi:10.1007/s00453-021-00831-w.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

8 Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. In *Proceedings of the 34th Conference on Artificial Intelligence (AAAI)*, pages 9810–9817, 2020. doi:10.1609/aaai.v34i06.6533.

9 Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.

10 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

11 Jessica Enright and Kitty Meeks. Deleting edges to restrict the size of an epidemic: a new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018. doi:10.1007/s00453-017-0311-7.

12 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021. doi:10.1016/j.jcss.2021.01.007.

13 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021. doi:10.1016/j.jcss.2020.08.001.

14 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. *Journal of Computer and System Sciences*, 119:1–18, 2021. doi:10.1016/j.jcss.2021.01.005.

15 Michael R Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

16 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020. doi:10.1016/j.tcs.2019.03.031.

17 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Delay-robust routes in temporal graphs, 2022. arXiv:2201.05390.

18 Eugen Füchsle, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Temporal connectivity: Coping with foreseen and unforeseen delays, 2022. To appear at SAND 2022. arXiv:2201.05011.

19 Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, 2015. doi:10.1140/epjb/e2015-60657-4.

20 Petter Holme and Jari Saramäki, editors. *Temporal Network Theory*. Springer, 2019. doi:10.1007/978-3-030-23495-9.

21 Richard M Karp. *Reducibility among combinatorial problems*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2_9.

22 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. doi:10.1006/jcss.2002.1829.

23 Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: Temporally disjoint paths. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 4090–4096. ijcai.org, 2021. doi:10.24963/ijcai.2021/563.

24 Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61, 2018. doi:10.1007/s13278-018-0537-7.

25 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016. doi:10.1080/15427951.2016.1177801.

26 Hendrik Molter, Malte Renken, and Philipp Zschoche. Temporal reachability minimization: Delaying vs. deleting. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 76:1–76:15, 2021. doi:10.4230/LIPIcs.MFCS.2021.76.

27 Manuel Sorge and Mathias Weller. The graph parameter hierarchy. unpublished manuscript, 2019. URL: https://manyu.pro/assets/parameter-hierarchy.pdf.

28 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. doi:10.1109/TKDE.2016.2594065.

29 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020. doi:10.1016/j.jcss.2019.07.006.

Maximally Satisfying Lower Quotas in the Hospitals/Residents Problem with Ties

Hiromichi Goko
Frontier Research Center, Toyota Motor Corporation, Aichi, Japan

Kazuhisa Makino
Research Institute for Mathematical Sciences, Kyoto University, Japan

Shuichi Miyazaki
Academic Center for Computing and Media Studies, Kyoto University, Japan

Yu Yokoi
Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan

Abstract

Motivated by the serious problem that hospitals in rural areas suffer from a shortage of residents, we study the Hospitals/Residents model in which hospitals are associated with lower quotas and the objective is to satisfy them as much as possible. When preference lists are strict, the number of residents assigned to each hospital is the same in any stable matching because of the well-known rural hospitals theorem; thus there is no room for algorithmic interventions. However, when ties are introduced to preference lists, this will no longer apply because the number of residents may vary over stable matchings.

In this paper, we formulate an optimization problem to find a stable matching with the maximum total satisfaction ratio for lower quotas. We first investigate how the total satisfaction ratio varies over choices of stable matchings in four natural scenarios and provide the exact values of these maximum gaps. Subsequently, we propose a strategy-proof approximation algorithm for our problem; in one scenario it solves the problem optimally, and in the other three scenarios, which are NP-hard, it yields a better approximation factor than that of a naive tie-breaking method. Finally, we show inapproximability results for the above-mentioned three NP-hard scenarios.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Algorithmic game theory

Keywords and phrases Stable matching, Hospitals/Residents problem, Lower quota, NP-hardness, Approximation algorithm, Strategy-proofness

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.31

Related Version *Full Version*: `https://arxiv.org/abs/2105.03093`

Funding This work was partially supported by the joint project of Kyoto University and Toyota Motor Corporation, titled “Advanced Mathematical Science for Mobility Society.”
Kazuhisa Makino: Supported by JSPS KAKENHI Grant Numbers JP19K22841, JP20H00609, and JP20H05967.
Shuichi Miyazaki: Supported by JSPS KAKENHI Grant Numbers JP20K11677 and JP16H02782.
Yu Yokoi: Supported by JSPS KAKENHI Grant Number JP18K18004 and JST PRESTO Grant Number JPMJPR212B.

Acknowledgements The authors thank the anonymous reviewers for their helpful comments.

1 Introduction

The Hospitals/Residents model (HR), a many-to-one matching model, has been extensively studied since the seminal work of Gale and Shapley [11]. Its input consists of a set of residents and a set of hospitals. Each resident has a preference over hospitals; similarly, each hospital has a preference over residents. In addition, each hospital is associated with a positive integer called the upper quota, which specifies the maximum number of residents it can accept. In this model, stability is the central solution concept, which requires the nonexistence of a blocking pair, i.e., a resident–hospital pair that has an incentive to deviate jointly from the current matching. In the basic model, each agent (resident or hospital) is assumed to have a strict preference for possible partners. For this model, the resident-oriented Gale–Shapley algorithm (also known as the deferred acceptance mechanism) is known to find a stable matching. This algorithm has advantages from both computational and strategic viewpoints: it runs in linear time and is strategy-proof for residents.

In reality, people typically have indifference among possible partners. Accordingly, a stable matching model that allows *ties* in preference lists, denoted by *HRT* in the context of HR, was introduced [20]. For such a model, several definitions of stability are possible. Among them, *weak stability* provides a natural concept, in which agents have no incentive to move within the ties. It is known that if we break the ties of an instance I arbitrarily, any stable matching of the resultant instance is a weakly stable matching of I. Hence, the Gale–Shapley algorithm can still be used to obtain a weakly stable matching. In applications, typically, ties are broken randomly, or participants are forced to report strict preferences even if their true preferences have ties. Hereafter, "stability" in the presence of ties refers to "weak stability," unless stated otherwise.

It is commonly known that HR plays an important role not only in theory but also in practice; for example, in assigning students to high schools [1,2] and residents to hospitals [30]. In such applications, "imbalance" is one of the major problems. For example, hospitals in urban areas are generally more popular than those in rural areas; hence it is likely that the former are well-staffed whereas the latter suffer from a shortage of doctors. One possible solution to this problem is to introduce a *lower quota* of each hospital, which specifies the minimum number of residents required by a hospital, and obtain a stable matching that satisfies both the upper and lower quotas. However, such a matching may not exist in general [16, 28], and determining if such a stable matching exists in HRT is known to be NP-complete (which is an immediate consequence from page 276 of [29]).

In general, it is too pessimistic to assume that a shortage of residents would force hospitals to go out of operation. In some cases, the hospital simply has to reduce its service level according to how much its lower quota is satisfied. In this scenario, a hospital will wish to satisfy the lower quota as much as possible, if not completely. To formulate this situation, we introduce the following optimization problem, which we call *HRT to Maximally Satisfy Lower Quotas* (*HRT-MSLQ*). Specifically, let R and H be the sets of residents and hospitals, respectively. All members in R and H have complete preference lists that may contain ties. Each hospital h has an upper quota $u(h)$, the maximum number of residents it can accept. The stability of a matching is defined with respect to these preference lists and upper quotas, as in conventional HRT. In addition, each hospital h is associated with a lower quota $\ell(h)$, which specifies the minimum number of residents required to keep its service level. We assume that $\ell(h) \le u(h) \le |R|$ for each $h \in H$. For a stable matching M, let $M(h)$ be the set of residents assigned to h. The *satisfaction ratio*, $s_M(h)$, of hospital $h \in H$ (with respect to $\ell(h)$) is defined as $s_M(h) = \min\left\{1, \frac{|M(h)|}{\ell(h)}\right\}$. Here, we let $s_M(h) = 1$ if $\ell(h) = 0$, because the lower quota is automatically satisfied in this case. The satisfaction ratio reflects a situation in which hospital h's service level increases linearly with respect to the number of residents up to $\ell(h)$ but does not increase after that, even though h is still willing to accept $u(h) - \ell(h)$ more residents. These $u(h) - \ell(h)$ positions may be considered as "marginal seats," which do not affect the service level but provide hospitals with advantages, such as generous work shifts. Our HRT-MSLQ problem asks us to maximize the total satisfaction ratio over the family $\mathcal{M}$ of all stable matchings in the problem instance, i.e.,

$$\max_{M\in\mathcal{M}} \sum_{h\in H} s_M(h).$$

The following are some remarks on our problem: (1) To our best knowledge, almost all previous works on lower quotas have investigated cases with no ties and have assumed lower quotas to be hard constraints. Refer to the discussion at the end of this section. (2) Our assumption that all preference lists are complete is theoretically a fundamental scenario used to study the satisfaction ratio for lower quotas. Moreover, there exist several cases in which this assumption is valid [4,14]. For example, according to Goto et al. [14], a complete list assumption is common in student–laboratory assignment in engineering departments of Japanese universities because it is mandatory that every student be assigned. (3) If preference lists contain no ties, the satisfaction ratio $s_M(h)$ is identical for any stable matching M because of the *rural hospitals theorem* [12,30,31]. Hence, there is no chance for algorithms to come into play if the stability is not relaxed. In our setting (i.e., with ties), the rural hospitals theorem implies that our task is essentially to find an optimal tie-breaking. However, it is unclear how to find such a tie-breaking.

Our Contributions. First, we study the goodness of any stable matching in terms of the total satisfaction ratios. For a problem instance I, let $\mathrm{OPT}(I)$ and $\mathrm{WST}(I)$, respectively, denote the maximum and minimum total satisfaction ratios of the stable matchings of I. For a family of problem instances $\mathcal{I}$, let $\Lambda(\mathcal{I}) = \max_{I\in\mathcal{I}} \frac{\mathrm{OPT}(I)}{\mathrm{WST}(I)}$ denote the maximum gap of the total satisfaction ratios. In this paper, we consider the following four fundamental scenarios of $\mathcal{I}$: (i) *general model*, which consists of all problem instances, (ii) *uniform model*, in which all hospitals have the same upper and lower quotas, (iii) *marriage model*, in which each hospital has an upper quota of 1 and a lower quota of either 0 or 1, and (iv) *R-side ML model*, in which all residents have identical preference lists. The exact values of $\Lambda(\mathcal{I})$ for all such fundamental scenarios are listed in the first row of Table 1, where $n = |R|$. In the uniform model, we write $\theta = \frac{u(h)}{\ell(h)}$ for the ratio of the upper and lower quotas, which is common to all hospitals. Further detailed analyses can be found in the full version [13].

Subsequently, we consider our problem algorithmically. Note that the aforementioned maximum gap corresponds to the worst-case approximation factor of the *arbitrarily tie-breaking Gale–Shapley algorithm*, which is frequently used in practice; this algorithm first breaks ties in the preference lists of agents arbitrarily and then applies the Gale–Shapley algorithm on the resulting preference lists. This correspondence easily follows from the rural hospitals theorem (see the full version [13] for the details).

In this paper, we show that there are two types of difficulties inherent in our problem HRT-MSLQ for all scenarios except (iv). Even for scenarios (i)–(iii), we show that (1) the problem is NP-hard and that (2) there is no algorithm that is strategy-proof for residents and always returns an optimal solution; see Section 6 and Appendix A.1.

We then consider strategy-proof approximation algorithms. We propose a strategy-proof algorithm Double Proposal, which is applicable in all above possible scenarios, whose approximation factor is substantially better than that of the arbitrary tie-breaking method. The approximation factors are listed in the second row of Table 1, where ϕ is a function defined by $\phi(1) = 1$, $\phi(2) = \frac{3}{2}$, and $\phi(n) = n(1 + \lfloor\frac{n}{2}\rfloor)/(n + \lfloor\frac{n}{2}\rfloor)$ for any $n \geq 3$. Note that $\frac{\theta^2+\theta-1}{2\theta-1} < \theta$ holds whenever $\theta > 1$. We also provide inapproximability results in the last row, where ϵ denotes an arbitrarily small positive constant.

Table 1 Maximum gap $\Lambda(\mathcal{I})$, approximation factor of Double Proposal, and inapproximability of HRT-MSLQ for four fundamental scenarios $\mathcal{I}$.

	General	Uniform	Marriage	R-side ML
Maximum gap $\Lambda(\mathcal{I})$ (i.e., Approx. factor of arbitrary tie-breaking GS)	$n+1$	θ	2	$n+1$
Approx. factor of Double Proposal	$\phi(n)$ $(\sim \frac{n+2}{3})$	$\frac{\theta^2+\theta-1}{2\theta-1}$	1.5	1
Inapproximability	$n^{\frac{1}{4}-\epsilon}$	$\frac{3\theta+4}{2\theta+4} - \epsilon$	$\frac{9}{8} - \epsilon$	—

$*$) Under P $\neq$ NP
†) Under the Unique Games Conjecture

Techniques. Our algorithm Double Proposal is based on the resident-oriented Gale–Shapley algorithm and is inspired by previous research on approximation algorithms [17, 25] for another NP-hard problem called MAX-SMTI. Unlike in the conventional Gale–Shapley algorithm, our algorithm allows each resident r to make proposals twice to each hospital. Among the hospitals in the top tie of the current preference list, r prefers hospitals to which r has not yet proposed to those which r has already proposed to once. When a hospital h receives a new proposal from r, hospital h may accept or reject it, and in the former case, h may reject a currently assigned resident to accommodate r. In contrast to the conventional Gale–Shapley algorithm, a rejection may occur even if h is not full. If at least $\ell(h)$ residents are currently assigned to h and at least one of them has not been rejected by h so far, then h rejects such a resident, regardless of its preference. This process can be considered as the algorithm dynamically finding a tie-breaking in r's preference list.

The main difficulty in our problem originates from the complicated form of our objective function $s(M) = \sum_{h \in H} \min\{1, \frac{|M(h)|}{\ell(h)}\}$. In particular, non-linearity of $s(M)$ makes the analysis of the approximation factor of Double Proposal considerably hard. We therefore introduce some new ideas and techniques to analyze the maximum gap Λ and approximation factor of our algorithm, which is one of the main novelties of this paper.

To estimate the approximation factor of the algorithm, we need to compare objective values of a stable matching M output by the algorithm and an (unknown) optimal stable matching N. A typical technique used to compare two matchings is to consider a graph of their union. In the marriage model, the connected components of the union are paths and cycles, both of which are easy to analyze; however, this is not the case in a general many-to-one matching model. For some problems, this approach still works via "cloning," which transforms an instance of HR into that of the marriage model by replacing each hospital h with an upper quota of $u(h)$ by $u(h)$ hospitals with an upper quota of 1. Unfortunately, however, in HRT-MSLQ there seems to be no simple way to transform the general model into the marriage model because of the non-linearity of the objective function.

In our analysis of the uniform model, the union graph of M and N may have a complex structure. We categorize hospitals using a procedure like breadth-first search starting from the set of hospitals h with the satisfaction ratio $s_N(h)$ larger than $s_M(h)$, which allows us to provide a tight bound on the approximation factor. For the general model, instead of using the union graph, we define two vectors that distribute the values $s(M)$ and $s(N)$ to the residents. By making use of the local optimality of M proven in Section 3, we compare such two vectors and give a tight bound on the approximation factor.

We finally remark that the improvement of Double Proposal over the maximum gap shows that our problem exhibits a different phenomenon from that of MAX-SMTI because the approximation factor of MAX-SMTI cannot be improved from a naive tie-breaking method if strategy-proofness is imposed [17].

Related Work. Recently, the Hospitals/Residents problems with lower quotas are quite popular in the literature; however, most of these studies are on settings without ties. The problems related to HRT-MSLQ can be classified into three models. The model by Hamada et al. [16], denoted by HR-LQ-2 in [28], is the closest to ours. The input of this model is the same as ours, but the hard and soft constraints are different from ours; their solution must satisfy both upper and lower quotas, the objective being to maximize the stability (e.g., to minimize the number of blocking pairs). Another model, introduced by Biró et al. [5] and denoted by HR-LQ-1 in [28], allows some hospitals to be closed; a closed hospital is not assigned any resident. They showed that it is NP-complete to determine the existence of a stable matching. This model was further studied by Boehmer and Heeger [6] from a parameterized complexity perspective. Huang [19] introduced the *classified stable matching* model, in which each hospital defines a family of subsets R of residents and each subset of R has an upper and lower quota. This model was extended by Fleiner and Kamiyama [9] to a many-to-many matching model where both sides have upper and lower quotas. Apart from these, several matching problems with lower quotas have been studied in the literature, whose solution concepts are different from stability [3, 10, 26, 27, 33].

Paper Organization. The rest of the paper is organized as follows. Section 2 formulates our problem HRT-MSLQ, and Section 3 describes our algorithm Double Proposal for HRT-MSLQ. Section 4 shows the strategy-proofness of Double Proposal. Section 5 is devoted to proving the maximum gaps Λ and approximation factors of algorithm Double Proposal for the several scenarios mentioned above. Finally, Section 6 provides hardness results such as NP-hardness and inapproximability for several scenarios. Because of space constraints, some proofs are omitted and included in the full version [13].

2 Problem Definition

Let $R = \{r_1, r_2, \ldots, r_n\}$ be a set of residents and $H = \{h_1, h_2, \ldots, h_m\}$ be a set of hospitals. Each hospital h has a lower quota $\ell(h)$ and an upper quota $u(h)$ such that $\ell(h) \leq u(h) \leq n$. We sometimes denote a hospital h's quota pair as $[\ell(h), u(h)]$ for simplicity. Each resident has a preference list over hospitals, which is complete and may contain ties. If a resident r prefers a hospital h_i to h_j, we write $h_i \succ_r h_j$. If r is indifferent between h_i and h_j (including the case that $h_i = h_j$), we write $h_i =_r h_j$. We use the notation $h_i \succeq_r h_j$ to signify that $h_i \succ_r h_j$ or $h_i =_r h_j$ holds. Similarly, each hospital has a preference list over residents and the same notations as above are used. In this paper, a preference list is denoted by one row, from left to right according to the preference order. When two or more agents are of equal preference, they are enclosed in parentheses. For example, "r_1: h_3 (h_2 h_4) h_1" is a preference list of resident r_1 such that h_3 is the top choice, h_2 and h_4 are the second choice with equal preference, and h_1 is the last choice.

An *assignment* is a subset of $R \times H$. For an assignment M and a resident r, let $M(r)$ be the set of hospitals h such that $(r, h) \in M$. Similarly, for a hospital h, let $M(h)$ be the set of residents r such that $(r, h) \in M$. An assignment M is called a *matching* if $|M(r)| \leq 1$ for each resident r and $|M(h)| \leq u(h)$ for each hospital h. For a matching M, a resident r

is called *matched* if $|M(r)| = 1$ and *unmatched* otherwise. If $(r, h) \in M$, we say that r is *assigned to* h and h is *assigned* r. We sometimes abuse notation $M(r)$ to denote the unique hospital where r is assigned. A hospital h is called *deficient* or *sufficient* if $|M(h)| < \ell(h)$ or $\ell(h) \leq |M(h)| \leq u(h)$, respectively. Additionally, a hospital h is called *full* if $|M(h)| = u(h)$ and *undersubscribed* otherwise.

A resident–hospital pair (r, h) is called a *blocking pair* for a matching M (or we say that (r, h) *blocks* M) if (i) r is either unmatched in M or prefers h to $M(r)$ and (ii) h is either undersubscribed in M or prefers r to at least one resident in $M(h)$. A matching is called *stable* if it admits no blocking pair. Recall that the satisfaction ratio of a hospital h (which is also called *the score* of h) in a matching M is defined by $s_M(h) = \min\{1, \frac{|M(h)|}{\ell(h)}\}$, where we define $s_M(h) = 1$ if $\ell(h) = 0$. The *total satisfaction ratio* (also called *the score*) of a matching M, is the sum of the scores of all hospitals, that is, $s(M) = \sum_{h \in H} s_M(h)$. The Hospitals/Residents problem with Ties to Maximally Satisfy Lower Quotas, denoted by *HRT-MSLQ*, is to find a stable matching M that maximizes the score $s(M)$. The optimal score of an instance I is denoted by $\mathrm{OPT}(I)$.

Note that if $|R| \geq \sum_{h \in H} u(h)$, then all hospitals are full in any stable matching (recall that preference lists are complete). Hence, all stable matchings have the same score $|H|$, and the problem is trivial. Therefore, throughout this paper, we assume $|R| < \sum_{h \in H} u(h)$. In this setting, all residents are matched in any stable matching as an unmatched resident forms a blocking pair with an undersubscribed hospital.

3 Algorithm

In this section, we present our algorithm Double Proposal for HRT-MSLQ along with a few of its basic properties. Its strategy-proofness and approximation factors for several models are presented in the following sections.

Our proposed algorithm Double Proposal is based on the resident-oriented Gale–Shapley algorithm but allows each resident r to make proposals twice to each hospital. Here, we explain the ideas underlying this modification.

Let us apply the ordinary resident-oriented Gale–Shapley algorithm to HRT-MSLQ, which starts with an empty matching $M := \emptyset$ and repeatedly updates M by a proposal-acceptance/rejection process. In each iteration, the algorithm takes a currently unassigned resident r and lets her propose to the hospital at the top of her current list. If the preference list of resident r contains ties, the proposal order of r depends on how to break the ties in her list. Hence, we need to define a priority rule for hospitals that are in a tie. Recall that our objective function is given by $s(M) = \sum_{h \in H} \min\{1, \frac{|M(h)|}{\ell(h)}\}$. This value immediately increases by $\frac{1}{\ell(h)}$ if r proposes to a deficient hospital h, whereas it does not increase if r proposes to a sufficient hospital h', although the latter may cause a rejection of some resident if h' is full. Therefore, a naive greedy approach is to let r first prioritize deficient hospitals over sufficient hospitals and then prioritize those with small lower quotas among deficient hospitals. This approach is useful for attaining a larger objective value for some instances; however, it is not enough to improve the approximation factor in the sense of worst case analysis, as a deficient hospital h in some iteration might become sufficient later and it might be better if r had made a proposal to a hospital other than h in the tie. Furthermore, this naive approach sacrifices strategy-proofness as demonstrated in Appendix A.2. This failure of strategy-proofness follows from the adaptivity of this tie-breaking rule, in the sense that the proposal order of each resident is affected by the other residents' behaviors.

In our algorithm Double Proposal, each resident can propose twice to each hospital. If the head of r's preference list is a tie when r makes a proposal, then the hospitals to which r has not yet proposed are prioritized. This idea was inspired by an algorithm of [17]. Recall that each hospital h has an upper quota $u(h)$ and a lower quota $\ell(h)$. In our algorithm, we use $\ell(h)$ as a dummy upper quota. Whenever $|M(h)| < \ell(h)$, a hospital h accepts any proposal. If h receives a new proposal from r when $|M(h)| \geq \ell(h)$, then h checks whether there is a resident in $M(h) \cup \{r\}$ who has not been rejected by h so far. If such a resident exists, h rejects that resident regardless of the preference of h. Otherwise, we apply the usual acceptance/rejection operation, i.e., h accepts r if $|M(h)| < u(h)$ and otherwise replaces r with the worst resident r' in $M(h)$. Roughly speaking, the first proposals are used to implement priority on deficient hospitals, and the second proposals are used to guarantee stability.

Formally, our algorithm Double Proposal is described in Algorithm 1. For convenience, in the preference list, a hospital h that is not included in any tie is regarded as a tie consisting of h only. We say that a resident is *rejected* by a hospital h if she is chosen as r' in Lines 12 or 17. To argue strategy-proofness, we need to make the algorithm deterministic. To this end, we remove arbitrariness using indices of agents as follows. If there are multiple hospitals (resp., residents) satisfying the condition to be chosen at Lines 5 or 7 (resp., at Lines 12 or 17), take the one with the smallest index (resp., with the largest index). Furthermore, when there are multiple unmatched residents at Line 3, take the one with the smallest index. In this paper, Double Proposal always refers to this deterministic version.

Algorithm 1 Double Proposal.

```
Input: An instance $I$ where each $h \in H$ has quotas $[\ell(h), u(h)]$.
Output: A stable matching $M$.
 1: $M := \emptyset$
 2: while there is an unmatched resident do
 3:   Let $r$ be any unmatched resident and $T$ be the top tie of $r$'s list.
 4:   if $T$ contains a hospital to which $r$ has not proposed yet then
 5:     Let $h$ be such a hospital with minimum $\ell(h)$.
 6:   else
 7:     Let $h$ be a hospital with minimum $\ell(h)$ in $T$.
 8:   end if
 9:   if $|M(h)| < \ell(h)$ then
10:     Let $M := M \cup \{(r, h)\}$.
11:   else if there is a resident in $M(h) \cup \{r\}$ who has not been rejected by $h$ then
12:     Let $r'$ be such a resident (possibly $r' = r$).
13:     Let $M := (M \cup \{(r, h)\}) \setminus \{(r', h)\}$.
14:   else if $|M(h)| < u(h)$ then
15:     $M := M \cup \{(r, h)\}$.
16:   else {i.e., when $|M(h)| = u(h)$ and all residents in $M(h) \cup \{r\}$ have been rejected by
      $h$ once}
17:     Let $r'$ be any resident that is worst in $M(h) \cup \{r\}$ for $h$ (possibly $r' = r$).
18:     Let $M := (M \cup \{(r, h)\}) \setminus \{(r', h)\}$.
19:     Delete $h$ from $r'$'s list.
20:   end if
21: end while
22: Output $M$ and halt.
```

▶ **Lemma 1.** *Algorithm* Double Proposal *runs in linear time and outputs a stable matching.*

Proof. Clearly, the size of the input is $O(|R||H|)$. As each resident proposes to each hospital at most twice, the while loop is iterated at most $2|R||H|$ times. At Lines 5 and 7, a resident prefers hospitals with smaller $\ell(h)$, and hence we need to sort hospitals in each tie in an increasing order of the values of ℓ. Since $0 \leq \ell(h) \leq n$ for each $h \in H$, ℓ has only $|R|+1$ possible values. Therefore, the required sorting can be done in $O(|R||H|)$ time as a preprocessing step using a method like bucket sort. Thus, our algorithm runs in linear time.

Observe that a hospital h is deleted from r's list only if h is full. Additionally, once h becomes full, it remains so afterward. Since each resident has a complete preference list and $|R| < \sum_{h \in H} u(h)$, the preference list of each resident never becomes empty. Therefore, all residents are matched in the output M.

Suppose, to the contrary, that M is not stable, i.e., there is a pair (r, h) such that (i) r prefers h to $M(r)$ and (ii) h is either undersubscribed or prefers r to at least one resident in $M(h)$. By the algorithm, (i) implies that r is rejected by h twice. Just after the second rejection, h is full, and all residents in $M(h)$ have once been rejected by h and are no worse than r for h. Since $M(h)$ is monotonically improving for h, at the end of the algorithm h is still full and no resident in $M(h)$ is worse than r, which contradicts (ii). ◀

In addition to stability, the output of Double Proposal satisfies the following property, which plays a key role in the analysis of the approximation factors in Section 5.

▶ **Lemma 2.** *Let M be the output of* Double Proposal*, r be a resident, and h and h' be hospitals such that $h =_r h'$ and $M(r) = h$. Then, we have the following conditions:*

(i) *If $\ell(h) > \ell(h')$, then $|M(h')| \geq \ell(h')$.*

(ii) *If $|M(h)| > \ell(h)$, then $|M(h')| \geq \ell(h')$.*

Proof. (i) Since $h =_r h'$, $\ell(h) > \ell(h')$, and r is assigned to h in M, the definition of the algorithm (Lines 4, 5, and 7) implies that r proposed to h' and was rejected by h' before she proposes to h. Just after this rejection occurred, $|M(h')| \geq \ell(h')$ holds. Since $|M(h')|$ is monotonically increasing, we also have $|M(h')| \geq \ell(h')$ at the end.

(ii) Since $|M(h)| > \ell(h)$, the value of $|M(h)|$ changes from $\ell(h)$ to $\ell(h)+1$ at some moment of the algorithm. By Line 11 of the algorithm, at any point after this, $M(h)$ consists only of residents who have once been rejected by h. Since $M(r) = h$ for the output M, at some moment r must have made the second proposal to h. By Line 4 of the algorithm, $h =_r h'$ implies that r has been rejected by h' at least once, which implies that $|M(h')| \geq \ell(h')$ at this moment and also at the end. ◀

Lemma 2 states some local optimality of Double Proposal. Suppose that we reassign r from h to h'. Then, h may lose and h' may gain score, but Lemma 2 says that the objective value does not increase. To see this, note that if the objective value were to increase, h' must gain score and h would either not lose score or lose less score than h' would gain. The former and the latter are the "if" parts of (ii) and (i), respectively, and in either case the conclusion $|M(h')| \geq \ell(h')$ implies that h' cannot gain score by accepting one more resident.

4 Strategy-proofness

An algorithm is called *strategy-proof* for residents if it gives residents no incentive to misrepresent their preferences. The precise definition follows. An algorithm that always outputs a matching deterministically can be regarded as a mapping from instances of HRT-MSLQ into matchings. Let A be an algorithm. We denote by $A(I)$ the matching returned by A for an instance I. For any instance I, let $r \in R$ be any resident, who has a preference $\succeq_r$. Additionally, let I' be an instance of HRT-MSLQ which is obtained from I by replacing $\succeq_r$ with some other $\succeq'_r$. Furthermore, let $M := A(I)$ and $M' := A(I')$. Then, A is strategy-proof if $M(r) \succeq_r M'(r)$ holds regardless of the choices of I, r, and $\succeq'_r$.

In the setting without ties, it is known that the resident-oriented Gale–Shapley algorithm is strategy-proof for residents (even if preference lists are incomplete) [8,15,32]. Furthermore, it has been proved that no algorithm can be strategy-proof for both residents and hospitals [32]. As in many existing papers on two-sided matching, we use the term "strategy-proofness" to refer to strategy-proofness for residents.

Before proving the strategy-proofness of Double Proposal, we remark that the exact optimization and strategy-proofness are incompatible even if a computational issue is set aside. The following fact is demonstrated in Appendix A.1.

▶ **Proposition 3.** *There is no algorithm that is strategy-proof for residents and returns an optimal solution for any instance of HRT-MSLQ. The statement holds even for the uniform and marriage models.*

This proposition implies that, if we require strategy-proofness for an algorithm, then we should consider approximation even in the absence of computational constraints. Now, we show the strategy-proofness of our approximation algorithm.

▶ **Theorem 4.** *Algorithm* Double Proposal *is strategy-proof for residents.*

Proof. To establish the strategy-proofness, we show that an execution of Double Proposal for an instance I can be described as an application of the resident-oriented Gale–Shapley algorithm to an auxiliary instance I^*. The construction of I^* is based on the proof of Lemma 8 in [17]; however, we need nontrivial extensions.

Let R and H be the sets of residents and hospitals in I, respectively. An auxiliary instance I^* is an instance of the Hospitals/Residents problem that has neither lower quotas nor ties and allows incomplete lists. The set of residents in I^* is $R' \cup D$, where $R' = \{r'_1, r'_2, \dots, r'_n\}$ is a copy of R and $D = \{\, d_{j,p} \mid j = 1, 2, \dots, m,\ p = 1, 2, \dots, u(h_j) \,\}$ is a set of $\sum_{j=1}^{m} u(h_j)$ dummy residents. The set of hospitals in I^* is $H^\circ \cup H^\bullet$, where each of $H^\circ = \{h^\circ_1, h^\circ_2, \dots, h^\circ_m\}$ and $H^\bullet = \{h^\bullet_1, h^\bullet_2, \dots, h^\bullet_m\}$ is a copy of H. Each hospital $h^\circ_j \in H^\circ$ has an upper quota $u(h_j)$ while each $h^\bullet_j \in H^\bullet$ has an upper quota $\ell(h_j)$.

For each resident $r'_i \in R'$, her preference list is defined as follows. Consider any tie $(h_{j_1} h_{j_2} \cdots h_{j_k})$ in r_i's preference list. Let $j'_1\, j'_2 \cdots j'_k$ be a permutation of $j_1\, j_2 \cdots j_k$ such that $\ell(h_{j'_1}) \le \ell(h_{j'_2}) \le \cdots \le \ell(h_{j'_k})$, and for each j'_p, j'_q with $\ell(h_{j'_p}) = \ell(h_{j'_q})$, $p < q$ implies $j'_p < j'_q$. We replace the tie $(h_{j_1} h_{j_2} \cdots h_{j_k})$ with a strict order of $2k$ hospitals $h^\bullet_{j'_1} h^\bullet_{j'_2} \cdots h^\bullet_{j'_k} h^\circ_{j'_1} h^\circ_{j'_2} \cdots h^\circ_{j'_k}$. The preference list of r'_i is obtained by applying this operation to all ties in r_i's list, where a hospital not included in any tie is regarded as a tie of length one. The following is an example of the correspondence between the preference lists of r_i and r'_i:

$$\begin{array}{lll} r_i: & (\ h_2\ h_4\ h_5\ )\ h_3\ (\ h_1\ h_6\ ) & \text{where} \quad \ell(h_4) = \ell(h_5) < \ell(h_2) \text{ and } \ell(h_6) < \ell(h_1) \\ r'_i: & h^\bullet_4\ h^\bullet_5\ h^\bullet_2\ h^\circ_4\ h^\circ_5\ h^\circ_2\ h^\bullet_3\ h^\circ_3\ h^\bullet_6\ h^\bullet_1\ h^\circ_6\ h^\circ_1 & \end{array}$$

For each $j = 1, 2, \dots, m$, the dummy residents $d_{j,p}$ $(p = 1, 2, \dots, u(h_j))$ have the same list:

$$d_{j,p}: \quad h^\circ_j \quad h^\bullet_j$$

For $j = 1, 2, \ldots, m$, let $P(h_j)$ be the preference list of h_j in I and let $Q(h_j)$ be the strict order on R' obtained by replacing residents r_i with r'_i and breaking ties so that residents in the same tie of $P(h_j)$ are ordered in ascending order of indices. The preference lists of hospitals h°_j and $h^\bullet_j$ are then defined as follows:

$$\begin{array}{lccc}
h^\circ_j: & Q(h_i) & d_{j,1}\ d_{j,2}\ \cdots\ d_{j,u(h_j)} & \\
h^\bullet_j: & d_{j,1}\ d_{j,2}\ \cdots\ d_{j,u(h_j)} & r'_1\ \ r'_2\ \ \cdots\ r'_n &
\end{array}$$

Let M be the output of Double Proposal applied to I. For each resident r_i, there are two cases: she has never been rejected by $M(r_i)$, and she had been rejected once by $M(r_i)$ and accepted upon her second proposal. Let M_1 be the set of pairs $(r_i, M(r_i))$ of the former case and M_2 be that of the latter. Note that $|M_1(h_j)| \le \ell(h_j)$ for any h_j. Define a matching M^* of I^* by

$$\begin{aligned}
M^* = &\{\, (r'_i, h^\circ_j) \mid (r_i, h_j) \in M_2 \,\} \cup \{\, (r'_i, h^\bullet_j) \mid (r_i, h_j) \in M_1 \,\} \\
&\cup \{\, (d_{j,p}, h^\circ_j) \mid \ 1 \le p \le u(h_j) - |M_2(h_j)| \ \} \\
&\cup \{\, (d_{j,p}, h^\bullet_j) \mid \ u(h_j) - |M_2(h_j)| < p \le \min\{u(h_j) - |M(h_j)| + \ell(h_j),\ u(h_j)\} \ \}\,.
\end{aligned}$$

Then, the following holds.

▶ **Lemma 5.** *M^* coincides with the output of the resident-oriented Gale–Shapley algorithm applied to the auxiliary instance I^*.*

We now complete the proof of the theorem.

Given an instance I, suppose that some resident r_i changes her preference list from $\succeq_{r_i}$ to some other $\succeq'_{r_i}$. Let J be the resultant instance. Define an auxiliary instance J^* from J in the manner described above. Let N be the output of Double Proposal for J and N^* be a matching defined from N as we defined M^* from M. By Lemma 5, the resident-oriented Gale–Shapley algorithm returns M^* and N^* for I^* and J^*, respectively. Note that all residents except r'_i have the same preference lists in I^* and J^* and so do all hospitals. Therefore, by the strategy-proofness of the Gale–Shapley algorithm, we have $M^*(r'_i) \succeq_{r'_i} N^*(r'_i)$. By the definitions of I^*, J^*, M^*, and N^*, we have $M(r_i) \succeq_{r_i} N(r_i)$, which means that r_i is no better off in N than in M with respect to her true preference $\succeq_{r_i}$. Thus, Double Proposal is strategy-proof for residents. ◀

5 Maximum Gaps and Approximation Factors of Double Proposal

In this section, we analyze the approximation factors of our algorithm, together with the maximum gaps Λ for the four models mentioned in Section 1. All results in this section are summarized in the first and second rows of Table 1 in Section 1.

For an instance I of HRT-MSLQ, let OPT(I) and WST(I) respectively denote the maximum and minimum scores over all stable matchings of I, and let ALG(I) be the score of the output of our algorithm Double Proposal. Then, WST(I) can be the score of the output of the algorithm that first breaks ties arbitrarily and then applies the Gale–Shapley algorithm for the resultant instance (see the full version [13]). Therefore, the maximum gap is equivalent to the approximation factor of such arbitrary tie-breaking GS algorithm.

For a model $\mathcal{I}$ (i.e., subfamily of problem instances of HRT-MSLQ), let

$$\Lambda(\mathcal{I}) = \max_{I\in\mathcal{I}} \frac{\mathrm{OPT}(I)}{\mathrm{WST}(I)} \quad \text{and} \quad \mathrm{APPROX}(\mathcal{I}) = \max_{I\in\mathcal{I}} \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}.$$

In subsequent subsections, we provide exact values of $\Lambda(\mathcal{I})$ and $\mathrm{APPROX}(\mathcal{I})$ for the four fundamental models. Recall our assumptions that preference lists are complete, $|R| < \sum_{h\in H} u(h)$, and $\ell(h) \le u(h) \le n$ for each $h \in H$.

5.1 General Model

Let $\mathcal{I}_{\mathrm{Gen}}$ denote the family of all instances of HRT-MSLQ, which we call the *general model*.

▶ **Proposition 6.** *The maximum gap for the general model satisfies* $\Lambda(\mathcal{I}_{\mathrm{Gen}}) = n + 1$. *Moreover, this equality holds even if residents have a master list, and preference lists of hospitals contain no ties.*

We next obtain the value of $\mathrm{APPROX}(\mathcal{I}_{\mathrm{Gen}})$. Recall that ϕ is a function of $n = |R|$ defined by $\phi(1) = 1$, $\phi(2) = \frac{3}{2}$, and $\phi(n) = n(1 + \lfloor \frac{n}{2} \rfloor)/(n + \lfloor \frac{n}{2} \rfloor)$ for $n \ge 3$.

▶ **Theorem 7.** *The approximation factor of* Double Proposal *for the general model satisfies* $\mathrm{APPROX}(\mathcal{I}_{\mathrm{Gen}}) = \phi(n)$.

We provide a full proof in the full version of the paper [13]. Here, we present the ideas to show the inequality $\frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)} \le \phi(n)$ for any $I \in \mathcal{I}_{\mathrm{Gen}}$.

Proof sketch of Theorem 7. Let M be the output of the algorithm and N be an optimal stable matching. We define vectors p_M and p_N on R, which distribute the scores to residents. For each $h \in H$, among residents in $M(h)$, we set $p_M(r) = \frac{1}{\ell(h)}$ for $\min\{\ell(h), |M(h)|\}$ residents and $p_M(r) = 0$ for the remaining $|M(h)| - \min\{\ell(h), |M(h)|\}$ residents. Similarly, we define p_N from N. We write $p_M(A) \coloneqq \sum_{r\in A} p_M(r)$ for any $A \subseteq R$. By definition, $p_M(M(h)) = s_M(h)$ and $p_N(N(h)) = s_N(h)$ for each $h \in H$, and hence $s(M) = \sum_{h\in H} s_M(h) = p_M(R)$ and $s(N) = \sum_{h\in H} s_N(h) = p_N(R)$. Thus, $\frac{p_N(R)}{p_M(R)} = \frac{s(N)}{s(M)}$, which needs to be bounded.

Let $R' = \{r'_1, r'_2, \dots, r'_n\}$ be a copy of R and identify p_N as a vector on R'. Consider a bipartite graph $G = (R, R'; E)$ whose edge set is $E \coloneqq \{\, (r_i, r'_j) \in R \times R' \mid p_M(r_i) \ge p_N(r'_j) \,\}$. For any matching $X \subseteq E$ in G, denote by $\partial(X) \subseteq R \cup R'$ the set of vertices covered by X. Then, $p_M(R \cap \partial(X)) \ge p_N(R' \cap \partial(X))$ holds since each edge $(r_i, r'_j) \in X \subseteq E$ satisfies $p_M(r_i) \ge p_N(r'_j)$. In addition, the value of $p_N(R' \setminus \partial(X)) - p_M(R \setminus \partial(X))$ is bounded from above by $|R \setminus \partial(X)| = |R| - |X| = n - |X|$ because $p_N(r') \le 1$ for any $r' \in R'$ and $p_M(r) \ge 0$ for any $r \in R$. Therefore, the existence of a matching $X \subseteq E$ with large $|X|$ helps us bound $\frac{p_N(R)}{p_M(R)}$. Indeed, the following claim plays a key role in our proof: ($\star$) The graph G admits a matching $X \subseteq E$ with $|X| \ge \lceil \frac{n}{2} \rceil$.

In the proof in the full version [13], the required bound of $\frac{p_N(R)}{p_M(R)}$ is obtained using a stronger version of ($\star$). Here we concentrate on showing ($\star$). To this end, we divide R into

$$\begin{aligned} R_+ &\coloneqq \{\, r \in R \mid M(r) \succ_r N(r) \,\}, \\ R_- &\coloneqq \{\, r \in R \mid N(r) \succ_r M(r) \text{ or } [M(r) =_r N(r),\ p_N(r) > p_M(r)] \,\}, \text{ and} \\ R_0 &\coloneqq \{\, r \in R \mid M(r) =_r N(r),\ p_M(r) \ge p_N(r) \,\}. \end{aligned}$$

Let R'_+, R'_-, R'_0 be the corresponding subsets of R'. We show the following two properties.

- There is an injection $\xi_+ \colon R_+ \to R'$ such that $p_M(r) = p_N(\xi_+(r))$ for every $r \in R_+$.
- There is an injection $\xi_- \colon R'_- \to R$ such that $p_N(r') = p_M(\xi_-(r'))$ for every $r' \in R'_-$.

We first define ξ_+. For each hospital h with $M(h) \cap R_+ \ne \emptyset$, there is $r \in M(h) \cap R_+$ with $h = M(r) \succ_r N(r)$. By the stability of N, hospital h is full in N. Then, we can define an injection $\xi_+^h \colon M(h) \cap R_+ \to N(h)$ so that $p_M(r) = p_N(\xi_+^h(r))$ for all $r \in M(h) \cap R_+$. By regarding $N(h)$ as a subset of R' and taking the direct sum of ξ_+^h for all hospitals h with $M(h) \cap R_+ \ne \emptyset$, we obtain a required injection $\xi_+ \colon R_+ \to R'$.

We next define ξ_-. For each hospital h' with $N(h') \cap R'_- \neq \emptyset$, any $r \in N(h') \cap R'_-$ satisfies either $h' = N(r) \succ_r M(r)$ or $[h' = N(r) =_r M(r),\ p_N(r) > p_M(r)]$. If some $r \in N(h') \cap R'_-$ satisfies the former, the stability of M implies that h' is full in M. If all $r \in N(h') \cap R'_-$ satisfy the latter, they all satisfy $0 \neq p_N(r) = \frac{1}{\ell(h')}$, and hence $|N(h') \cap R'_-| \leq \ell(h')$. Additionally, $p_N(r) > p_M(r)$ implies either $p_M(r) = 0$ or $\ell(h') < \ell(h)$, where $h \coloneqq M(r)$. Observe that $p_M(r) = 0$ implies $|M(h)| > \ell(h)$. By Lemma 2, each of $\ell(h') < \ell(h)$ and $|M(h)| > \ell(h)$ implies $|M(h')| \geq \ell(h') \geq |N(h') \cap R'_-|$. Then, in any case, we can define an injection $\xi_-^{h'} \colon N(h') \cap R'_- \to M(h')$ such that $p_N(r') = p_M(\xi_-^{h'}(r'))$ for all $r' \in N(h') \cap R'_-$. By taking the direct sum of $\xi_-^{h'}$ for all hospitals h' with $M(h') \cap R_- \neq \emptyset$, we obtain $\xi_- \colon R'_- \to R$.

Let $G^* = (R, R'; E^*)$ be a bipartite graph (possibly with multiple edges), where E^* is the disjoint union of E_+, E_-, and E_0, defined by

$$E_+ \coloneqq \{\, (r, \xi_+(r)) \mid r \in R_+ \,\}, \quad E_- \coloneqq \{\, (\xi_-(r'), r') \mid r \in R'_- \,\}, \text{ and}$$
$$E_0 \coloneqq \{\, (r, r') \mid r \in R_0 \text{ and } r' \text{ is the copy of } r \,\}.$$

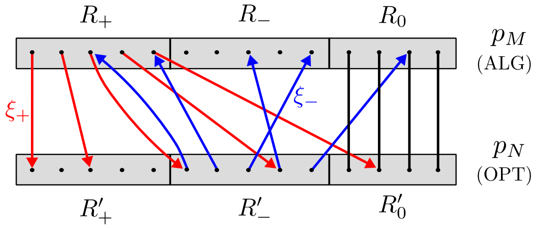


Figure 1 A graph $G^* = (R, R'; E^*)$.

See Fig. 1 for an example. By the definitions of ξ_+, ξ_-, and R_0, any edge (r, r') in E^* belongs to E, and hence any matching in G^* is also a matching in G. Since $\xi_+ \colon R_+ \to R'$ and $\xi_- \colon R'_- \to R$ are injections, we observe that every vertex in G^* is incident to at most two edges in E^*. Then, E^* is decomposed into paths and cycles, and hence E^* contains a matching of size at least $\lceil \frac{|E^*|}{2} \rceil$. Since $|E^*| = |R_+| + |R_-| + |R_0| = n$, this means that there exists a matching $X \subseteq E$ with $|X| \geq \lceil \frac{n}{2} \rceil$, as required. ◀

5.2 Uniform Model

Let $\mathcal{I}_{\text{Uniform}}$ denote the family of uniform problem instances of HRT-MSLQ, where an instance is called *uniform* if upper and lower quotas are uniform. In the rest of this subsection, we assume that ℓ and u are nonnegative integers to represent the common lower and upper quotas, respectively, and let $\theta \coloneqq \frac{u}{\ell}$ (≥ 1). We call $\mathcal{I}_{\text{Uniform}}$ the *uniform model.*

▶ **Proposition 8.** *The maximum gap for the uniform model satisfies* $\Lambda(\mathcal{I}_{\text{Uniform}}) = \theta$. *Moreover, this equality holds even if preference lists of hospitals contain no ties.*

▶ **Theorem 9.** *The approximation factor of* Double Proposal *for the uniform model satisfies* $\text{APPROX}(\mathcal{I}_{\text{uniform}}) = \frac{\theta^2+\theta-1}{2\theta-1}$.

Note that $\frac{\theta^2+\theta-1}{2\theta-1} < \theta$ whenever $\ell < u$ because $\theta - \frac{\theta^2+\theta-1}{2\theta-1} = \frac{(\theta-1)^2}{2\theta-1} > 0$. Here is the ideas to show that $\frac{\text{OPT}(I)}{\text{ALG}(I)} \leq \frac{\theta^2+\theta-1}{2\theta-1}$ holds for any $I \in \mathcal{I}_{\text{Uniform}}$.

Proof sketch of Theorem 9. Let M be the output of the algorithm and N be an optimal stable matching, and assume $s(M) < s(N)$. Consider a bipartite graph $(R, H; M \cup N)$, which may have multiple edges. Take an arbitrary connected component, and let R^* and H^* be the sets of residents and hospitals, respectively, contained in it. It is sufficient to bound $\frac{s_N(H^*)}{s_M(H^*)}$.

Let H_0 be the set of all hospitals in H^* having strictly larger scores in N than in M, i.e.,

$$H_0 \coloneqq \{\, h \in H^* \mid s_N(h) > s_M(h) \,\}.$$

Using this, we sequentially define

$$R_0 \coloneqq \{\, r \in R^* \mid N(r) \in H_0 \,\}, \quad H_1 \coloneqq \{\, h \in H^* \setminus H_0 \mid \exists r \in R_0 : M(r) = h \,\},$$
$$R_1 \coloneqq \{\, r \in R^* \mid N(r) \in H_1 \,\}, \quad H_2 \coloneqq H^* \setminus (H_0 \cup H_1), \quad \text{and} \quad R_2 \coloneqq R^* \setminus (R_0 \cup R_1).$$

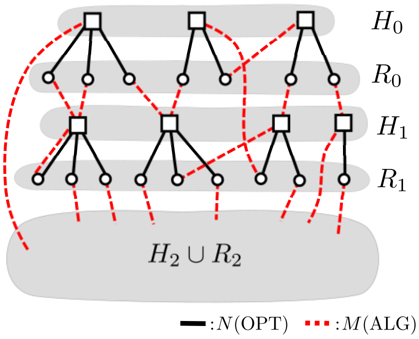


Figure 2 Example with $[\ell, u] = [2, 3]$.

See Fig. 2 for an example. We use scaled score functions $v_M \coloneqq \ell \cdot s_M$ and $v_N \coloneqq \ell \cdot s_N$ and write $v_M(A) = \sum_{h \in A} v_M(h)$ for any $A \subseteq H$. We bound $\frac{v_N(H^*)}{v_M(H^*)}$, which equals $\frac{s_N(H^*)}{s_M(H^*)}$. Note that the set of residents assigned to H^* is R^* in both M and N. The scores differ depending on how efficiently those residents are assigned. In this sense, we may think that a hospital h is assigned residents "efficiently" in M if $|M(h)| \le \ell$ and is assigned "most redundantly" if $|M(h)| = u$. Since $v_M(h) = \min\{\ell, |M(h)|\}$, we have $v_M(h) = |M(h)|$ in the former case and $v_M(h) = \frac{1}{\theta} \cdot |M(h)|$ in the latter. We show that hospitals in H_1 provide us with advantage of M; any hospital in H_1 is assigned residents either efficiently in M or most redundantly in N.

For any $h \in H_0$, $s_M(h) < s_N(h)$ implies $|M(h)| < \ell$. Then, the stability of M implies $M(r) \succeq_r N(r)$ for any $r \in R_0$. Hence, the following $\{H_1^{\succ}, H_1^{=}\}$ defines a bipartition of H_1:

$$H_1^{\succ} \coloneqq \{\, h \in H_1 \mid \exists r \in M(h) \cap R_0 : h \succ_r N(r) \,\},$$
$$H_1^{=} \coloneqq \{\, h \in H_1 \mid \forall r \in M(h) \cap R_0 : h =_r N(r) \,\}.$$

For each $h \in H_1^{\succ}$, as some r satisfies $h \succ_r N(r)$, the stability of N implies that h is full, i.e., h is assigned residents most redundantly, in N. Note that any $h \in H_1^{\succ}$ satisfies $v_M(h) \ge v_N(h)$ because $h \notin H_0$, and hence $v_M(h) = v_N(h) = \ell$. Then, $|N(h)| = u = \theta \cdot v_N(h) = (\theta - 1) \cdot v_M(h) + v_N(h)$ for each $h \in H_1^{\succ}$. Additionally, for any $h \in H^*$, we have $|N(h)| \ge \min\{\ell, |N(h)|\} = v_N(h)$. Since $|R^*| = \sum_{h \in H^*} |N(h)|$, we have

$$|R^*| \ge (\theta - 1) \cdot v_M(H_1^{\succ}) + v_N(H_1^{\succ}) + v_N(H^* \setminus H_1^{\succ}) = (\theta - 1) \cdot v_M(H_1^{\succ}) + v_N(H^*).$$

For each $h \in H_1^=$, there is $r \in R_0$ with $M(r) = h =_r N(r)$. As $r \in R_0$, the hospital $h' := N(r)$ belongs to H_0, and hence $|M(h')| < \ell$. Then, Lemma 2(ii) implies $|M(h)| \leq \ell$, i.e., h is assigned residents efficiently in M. Note that any $h \in H_0$ satisfies $v_M(h) < v_N(h) \leq \ell$. Then, the number of residents assigned to $H_0 \cup H_1^=$ is $v_M(H_0 \cup H_1^=)$. Additionally, the number of residents assigned to $H_1^\succ \cup H_2$ is at most $\theta \cdot v_M(H_1^\succ \cup H_2)$. Thus, we have

$$|R^*| \leq v_M(H_0 \cup H_1^=) + \theta \cdot v_M(H_1^\succ \cup H_2) = v_M(H^*) + (\theta - 1) \cdot v_M(H_1^\succ \cup H_2).$$

From these two estimations of $|R^*|$, we obtain $v_N(H^*) \leq (\theta - 1) \cdot v_M(H_2) + v_M(H^*)$, which gives us a relationship between $v_M(H^*)$ and $v_N(H^*)$. Combining this with other inequalities, we can obtain the required upper bound of $\frac{v_N(H^*)}{v_M(H^*)}$. ◀

5.3 Marriage Model

Let $\mathcal{I}_{\text{Marriage}}$ denote the family of instances of HRT-MSLQ, in which each hospital has an upper quota of 1. We call $\mathcal{I}_{\text{Marriage}}$ the *marriage model*. By definition, $[\ell(h), u(h)]$ in this model is either $[0, 1]$ or $[1, 1]$ for each $h \in H$. Since this is a one-to-one matching model, the union of two stable matchings can be partitioned into paths and cycles. By applying standard arguments used in other stable matching problems, we can obtain $\Lambda(\mathcal{I}_{\text{Marriage}}) = 2$ and $\text{APPROX}(\mathcal{I}_{\text{Marriage}}) = 1.5$.

As shown in Example 15 in Appendix A.1, there is no strategy-proof algorithm that can achieve an approximation factor better than 1.5 even in the marriage model. Therefore, we cannot improve this ratio without sacrificing strategy-proofness.

5.4 Resident-side Master List Model

Let $\mathcal{I}_{\text{R-ML}}$ denote the family of instances of HRT-MSLQ in which all residents have the same preference list. This is well studied in literature on stable matching [7, 21–23]. We call $\mathcal{I}_{\text{R-ML}}$ the *R-side ML model*. We have already shown in Proposition 6 that $\Lambda(\mathcal{I}_{\text{R-ML}}) = n + 1$. Our algorithm, however, solves this model exactly.

Note that this is not the case for the hospital-side master list model, which is NP-hard as shown in Theorem 14 below. This difference highlights the asymmetry of two sides in HRT-MSLQ.

6 Hardness Results

We obtain various hardness and inapproximability results for HRT-MSLQ. First, we show that HRT-MSLQ in the general model is inapproximable and that we cannot hope for a constant factor approximation.

▶ **Theorem 10.** *HRT-MSLQ is inapproximable within a ratio $n^{\frac{1}{4}-\epsilon}$ for any $\epsilon > 0$ unless P=NP.*

Proof. We show the theorem by way of a couple of reductions, one from the maximum independent set problem (*MAX-IS*) to the maximum 2-independent set problem (*MAX-2-IS*), and the other from MAX-2-IS to HRT-MSLQ.

For an undirected graph $G = (V, E)$, a subset $S \subseteq V$ is an *independent set* of G if no two vertices in S are adjacent. S is a *2-independent set* of G if the distance between any two vertices in S is at least 3. MAX-IS (resp. MAX-2-IS) asks to find an independent set (resp. 2-independent set) of maximum size. Let us denote by $\text{IS}(G)$ and $\text{IS}_2(G)$, respectively, the sizes of optimal solutions of MAX-IS and MAX-2-IS for G. We assume without loss of generality that input graphs are connected. It is known that, unless P=NP, there is no polynomial-time algorithm, given a graph $G_1 = (V_1, E_1)$, to distinguish between the two cases $\mathrm{IS}(G_1) \le |V_1|^{\epsilon_1}$ and $\mathrm{IS}(G_1) \ge |V_1|^{1-\epsilon_1}$, for any constant $\epsilon_1 > 0$ [34].

Now, we give the first reduction, which is based on the NP-hardness proof of the minimum maximal matching problem [18]. Let $G_1 = (V_1, E_1)$ be an instance of MAX-IS. We construct an instance $G_2 = (V_2, E_2)$ of MAX-2-IS as $V_2 = V_1 \cup E_1 \cup \{s\}$ and $E_2 = \{\, (v, e) \mid v \in V_1,\ e \in E_1, e \text{ is incident to } v \text{ in } G_1 \,\} \cup \{\, (s, e) \mid e \in E_1 \,\}$, where s is a new vertex not in $V_1 \cup E_1$. For any two vertices u and v in V_1, if their distance in G_1 is at least 2 then that in G_2 is at least 4. Hence, any independent set in G_1 is also a 2-independent set in G_2. Conversely, for any 2-independent set S in G_2, $S \cap V_1$ is independent in G_1 and $|S \cap (V_2 \setminus V_1)| \le 1$. These facts imply that $\mathrm{IS}_2(G_2)$ is either $\mathrm{IS}(G_1)$ or $\mathrm{IS}(G_1) + 1$. Since $|E_2| = 3|E_1| \le 3|V_1|^2$, distinguishing between $\mathrm{IS}_2(G_2) \le |E_2|^{\epsilon_2}$ and $\mathrm{IS}_2(G_2) \ge |E_2|^{1/2-\epsilon_2}$ for some constant $\epsilon_2 > 0$ would imply distinguishing between $\mathrm{IS}(G_1) \le |V_1|^{\epsilon_1}$ and $\mathrm{IS}(G_1) \ge |V_1|^{1-\epsilon_1}$ for some constant $\epsilon_1 > 0$, which in turn implies P=NP.

We then proceed to the second reduction. Let $G_2 = (V_2, E_2)$ be an instance of MAX-2-IS. Let $n_2 = |V_2|$, $m_2 = |E_2|$, $V_2 = \{v_1, v_2, \dots, v_{n_2}\}$, and $E_2 = \{e_1, e_2, \dots, e_{m_2}\}$. We construct an instance I of HRT-MSLQ as follows. For an integer p which will be determined later, define the set of residents of I as $R = \{\, r_{i,j} \mid 1 \le i \le n_2,\ 1 \le j \le p \,\}$, where $r_{i,j}$ corresponds to the jth copy of vertex $v_i \in V_2$. Next, define the set of hospitals of I as $H \cup Y$, where $H = \{\, h_k \mid 1 \le k \le m_2 \,\}$ and $Y = \{\, y_{i,j} \mid 1 \le i \le n_2,\ 1 \le j \le p \,\}$. The hospital h_k corresponds to the edge $e_k \in E_2$ and the hospital $y_{i,j}$ corresponds to the resident $r_{i,j}$.

We complete the reduction by giving preference lists and quotas in Fig. 3, where $1 \le i \le n_2$, $1 \le j \le p$, and $1 \le k \le m_2$. Here, $N(v_i) = \{\, h_k \mid e_k \text{ is incident to } v_i \text{ in } G_2 \,\}$ and "($N(v_i)$)" denotes the tie consisting of all hospitals in $N(v_i)$. Similarly, $N(e_k) = \{\, r_{i,j} \mid e_k \text{ is incident to } v_i \text{ in } G_2,\ 1 \le j \le p \,\}$ and "($N(e_k)$)" is the tie consisting of all residents in $N(e_k)$. The notation "$\cdots$" denotes an arbitrary strict order of all agents missing in the list.

$$
\begin{array}{llll}
r_{i,j}\colon & (\ N(v_i)\ )\quad y_{i,j}\quad \cdots \qquad\qquad & h_k\ [0,p]\colon & (\ N(e_k)\ )\quad \cdots \\
 & & y_{i,j}\ [1,1]\colon & r_{i,j}\qquad\qquad \cdots
\end{array}
$$

Figure 3 Preference lists of residents and hospitals.

We will show that $\mathrm{OPT}(I) = m_2 + p \cdot \mathrm{IS}_2(G_2)$. To do so, we first see a useful property. Let $G_3 = (V_3, E_3)$ be the subdivision graph of G_2, i.e., $V_3 = V_2 \cup E_2$ and $E_3 = \{\, (v, e) \mid v \in V_2, e \in E_2, e \text{ is incident to } v \text{ in } G_2 \,\}$. Then, the family $\mathcal{I}_2(G_2)$ of 2-independent sets in G_2 is characterized as follows [18]:

$$\mathcal{I}_2(G_2) = \left\{ V_2 \setminus \bigcup_{e \in M} \{\text{endpoints of } e\} \;\middle|\; M \text{ is a maximal matching of } G_3 \right\}.$$

In other words, for a maximal matching M of G_3, if we remove all vertices matched in M from V_2, then the remaining vertices form a 2-independent set of G_2, and conversely, any 2-independent set of G_2 can be obtained in this manner for some maximal matching M of G_3.

Let S be an optimal solution of G_2 in MAX-2-IS, i.e., a 2-independent set of size $\mathrm{IS}_2(G_2)$. Let $\tilde{M}$ be a maximal matching of G_3 corresponding to S. We construct a matching M of I as $M = M_1 \cup M_2$, where $M_1 = \{\, (r_{i,j}, h_k) \mid (v_i, e_k) \in \tilde{M},\ 1 \le j \le p \,\}$ and $M_2 = \{\, (r_{i,j}, y_{i,j}) \mid v_i \in S,\ 1 \le j \le p \,\}$. It is not hard to see that each resident is matched by exactly one of M_1 and M_2 and that no hospital exceeds its upper quota.

We then show the stability of M. Each resident matched by M_1 is assigned to a first-choice hospital, so if there were a blocking pair, then it would be of the form $(r_{i,j}, h_k)$ where $M(r_{i,j}) = y_{i,j}$ and $h_k \in N(v_i)$. Then, v_i is unmatched in $\tilde{M}$. Additionally, all residents assigned to h_k (if any) are its first choice; hence, h_k must be undersubscribed in M. Then, e_k is unmatched in $\tilde{M}$. $h_k \in N(v_i)$ implies that there is an edge $(v_i, e_k) \in E_3$, so $\tilde{M} \cup \{(v_i, e_k)\}$ is a matching of G_3, contradicting the maximality of $\tilde{M}$. Hence, M is stable in I.

A hospital in H has a lower quota of 0, so it obtains a score of 1. The number of hospitals in Y that are assigned a resident is $|M_2| = p|S| = p \cdot \mathrm{IS}_2(G_2)$. Hence, $s(M) = m_2 + p \cdot \mathrm{IS}_2(G_2)$. Therefore, we have $\mathrm{OPT}(I) \geq s(M) = m_2 + p \cdot \mathrm{IS}_2(G_2)$.

Conversely, let M be an optimal solution for I, i.e., a stable matching of score $\mathrm{OPT}(I)$. Note that each $r_{i,j}$ is assigned to a hospital in $N(v_i) \cup \{y_{i,j}\}$ as otherwise $(r_{i,j}, y_{i,j})$ blocks M. We construct a bipartite multi-graph $G_M = (V_2, E_2; F)$ where $V_2 = \{v_1, v_2, \ldots, v_{n_2}\}$ and $E_2 = \{e_1, e_2, \ldots, e_{m_2}\}$ are identified as vertices and edges of G_2, respectively, and an edge $(v_i, e_k)_j \in F$ if and only if $(r_{i,j}, h_k) \in M$. Here, a subscript j of edge $(v_i, e_k)_j$ is introduced to distinguish the multiplicity of edge (v_i, e_k). The degree of each vertex of G_M is at most p, so by Kőnig's edge coloring theorem [24], G_M is p-edge colorable and each color class c induces a matching M_c $(1 \leq c \leq p)$ of G_M. Each M_c is a matching of G_3, and by the stability of M, we can show that it is in fact a maximal matching of G_3. Let M_* be a minimum cardinality one among them.

Define a subset S of V_2 by removing vertices that are matched in M_* from V_2. By the above observation, S is a 2-independent set of G_2. We will bound its size. Note that $s(M) = \mathrm{OPT}(I)$ and each hospital in H obtains the score of 1, so M assigns residents to $\mathrm{OPT}(I) - m_2$ hospitals in Y and each such hospital receives one resident. There are pn_2 residents in total, among which $\mathrm{OPT}(I) - m_2$ ones are assigned to hospitals in Y, so the remaining $pn_2 - (\mathrm{OPT}(I) - m_2)$ ones are assigned to hospitals in H. Thus F contains this number of edges and so $|M_*| \leq \frac{pn_2 - (\mathrm{OPT}(I) - m_2)}{p} = n_2 - \frac{\mathrm{OPT}(I) - m_2}{p}$. Since $|V_2| = n_2$ and exactly one endpoint of each edge in M_* belongs to V_2, we have that $|S| = |V_2| - |M_*| \geq \frac{\mathrm{OPT}(I) - m_2}{p}$. Therefore $\mathrm{IS}_2(G_2) \geq |S| \geq \frac{\mathrm{OPT}(I) - m_2}{p}$. Hence, we obtain $\mathrm{OPT}(I) = m_2 + p \cdot \mathrm{IS}_2(G_2)$ as desired. Now we let $p = m_2$, and have $\mathrm{OPT}(I) = m_2(1 + \mathrm{IS}_2(G_2))$.

Therefore distinguishing between $\mathrm{OPT}(I) \leq (m_2)^{1+\delta}$ and $\mathrm{OPT}(I) \geq (m_2)^{3/2-\delta}$ for some δ would distinguish between $\mathrm{IS}_2(G_2) \leq (m_2)^{\epsilon_2}$ and $\mathrm{IS}_2(G_2) \geq (m_2)^{1/2-\epsilon_2}$ for some constant $\epsilon_2 > 0$. Since $n = |R| = n_2 m_2 \leq (m_2)^2$, a polynomial-time $n^{1/4-\epsilon}$-approximation algorithm for HRT-MSLQ can distinguish between the above two cases for a constant $\delta < \epsilon/2$. Hence, the existence of such an algorithm implies P=NP. This completes the proof. ◀

We then show inapproximability results for the uniform model and the marriage model under the Unique Games Conjecture (UGC).

▶ **Theorem 11.** *Under UGC, HRT-MSLQ in the uniform model is not approximable within a ratio* $\frac{3\theta+3}{2\theta+4} - \epsilon$ *for any positive* ϵ.

▶ **Theorem 12.** *Under UGC, HRT-MSLQ in the marriage model is not approximable within a ratio* $\frac{9}{8} - \epsilon$ *for any positive* ϵ.

Furthermore, we give two examples showing that HRT-MSLQ is NP-hard even in very restrictive settings. The first is a marriage model for which ties appear in one side only.

▶ **Theorem 13.** *HRT-MSLQ in the marriage model is NP-hard even if there is a master preference list of hospitals and ties appear only in preference lists of residents or only in preference lists of hospitals.*

The other is a setting like the capacitated house allocation problem, where all hospitals are indifferent among residents.

▶ **Theorem 14.** *HRT-MSLQ in the uniform model is NP-hard even if all the hospitals quotas are* $[1, 2]$*, preferences lists of all residents are strict, and all hospitals are indifferent among all residents (i.e., there is a master list of hospitals consisting of a single tie).*

References

1 Atila Abdulkadiroğlu, Parag A. Pathak, and Alvin E. Roth. The New York city high school match. *American Economic Review*, 95(2):364–367, 2005.

2 Atila Abdulkadiroğlu, Parag A. Pathak, Alvin E. Roth, and Tayfun Sönmez. The Boston public school match. *American Economic Review*, 95(2):368–371, 2005.

3 Ashwin Arulselvan, Ágnes Cseh, Martin Groß, David F. Manlove, and Jannik Matuschke. Matchings with lower quotas: Algorithms and complexity. *Algorithmica*, 80(1):185–208, 2018. doi:10.1007/s00453-016-0252-6.

4 Itai Ashlagi, Amin Saberi, and Ali Shameli. Assignment mechanisms under distributional constraints. *Oper. Res.*, 68(2):467–479, 2020. doi:10.1287/opre.2019.1887.

5 Péter Biró, Tamás Fleiner, Robert W. Irving, and David F. Manlove. The College Admissions problem with lower and common quotas. *Theor. Comput. Sci.*, 411(34-36):3136–3153, 2010. doi:10.1016/j.tcs.2010.05.005.

6 Niclas Boehmer and Klaus Heeger. A fine-grained view on stable many-to-one matching problems with lower and upper quotas. In Xujin Chen, Nikolai Gravin, Martin Hoefer, and Ruta Mehta, editors, *Web and Internet Economics - 16th International Conference, WINE 2020, Beijing, China, December 7-11, 2020, Proceedings*, volume 12495 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2020. doi:10.1007/978-3-030-64946-3_3.

7 Robert Bredereck, Klaus Heeger, Dusan Knop, and Rolf Niedermeier. Multidimensional stable roommates with master list. In Xujin Chen, Nikolai Gravin, Martin Hoefer, and Ruta Mehta, editors, *Web and Internet Economics - 16th International Conference, WINE 2020, Beijing, China, December 7-11, 2020, Proceedings*, volume 12495 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2020. doi:10.1007/978-3-030-64946-3_5.

8 Lester E. Dubins and David A. Freedman. Machiavelli and the Gale-Shapley algorithm. *The American Mathematical Monthly*, 88(7):485–494, 1981.

9 Tamás Fleiner and Naoyuki Kamiyama. A matroid approach to stable matchings with lower quotas. *Math. Oper. Res.*, 41(2):734–744, 2016. doi:10.1287/moor.2015.0751.

10 Daniel Fragiadakis, Atsushi Iwasaki, Peter Troyan, Suguru Ueda, and Makoto Yokoo. Strategyproof matching with minimum quotas. *ACM Trans. Economics and Comput.*, 4(1):6:1–6:40, 2015. doi:10.1145/2841226.

11 David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

12 David Gale and Marilda Sotomayor. Some remarks on the stable matching problem. *Discret. Appl. Math.*, 11(3):223–232, 1985. doi:10.1016/0166-218X(85)90074-5.

13 Hiromichi Goko, Kazuhisa Makino, Shuichi Miyazaki, and Yu Yokoi. Maximally satisfying lower quotas in the hospitals/residents problem with ties. *CoRR*, abs/2105.03093, 2021. arXiv:2105.03093.

14 Masahiro Goto, Atsushi Iwasaki, Yujiro Kawasaki, Ryoji Kurata, Yosuke Yasuda, and Makoto Yokoo. Strategyproof matching with regional minimum and maximum quotas. *Artificial Intelligence*, 235:40–57, 2016.

15 Dan Gusfield and Robert W. Irving. *The Stable marriage problem - structure and algorithms*. Foundations of computing series. MIT Press, 1989.

16 Koki Hamada, Kazuo Iwama, and Shuichi Miyazaki. The Hospitals/Residents problem with lower quotas. *Algorithmica*, 74(1):440–465, 2016. doi:10.1007/s00453-014-9951-z.

17 Koki Hamada, Shuichi Miyazaki, and Hiroki Yanagisawa. Strategy-proof approximation algorithms for the stable marriage problem with ties and incomplete lists. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China*, volume 149 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.9.

18 Joseph D. Horton and Kyriakos Kilakos. Minimum edge dominating sets. *SIAM J. Discret. Math.*, 6(3):375–387, August 1993.

19 Chien-Chung Huang. Classified stable matching. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1235–1253. SIAM, 2010. doi:10.1137/1.9781611973075.99.

20 Robert W. Irving. Stable marriage and indifference. *Discret. Appl. Math.*, 48(3):261–272, 1994. doi:10.1016/0166-218X(92)00179-P.

21 Robert W. Irving, David F. Manlove, and Sandy Scott. The stable marriage problem with master preference lists. *Discret. Appl. Math.*, 156(15):2959–2977, 2008. doi:10.1016/j.dam.2008.01.002.

22 Naoyuki Kamiyama. Stable matchings with ties, master preference lists, and matroid constraints. In Martin Hoefer, editor, *Algorithmic Game Theory - 8th International Symposium, SAGT 2015, Saarbrücken, Germany, September 28-30, 2015, Proceedings*, volume 9347 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2015. doi:10.1007/978-3-662-48433-3_1.

23 Naoyuki Kamiyama. Many-to-many stable matchings with ties, master preference lists, and matroid constraints. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 583–591. International Foundation for Autonomous Agents and Multiagent Systems, 2019. URL: http://dl.acm.org/citation.cfm?id=3331743.

24 Dénes Kőnig. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische*, 77(4):453–465, 1916.

25 Zoltán Király. Linear time local approximation algorithm for maximum stable marriage. *Algorithms*, 6(3):471–484, 2013. doi:10.3390/a6030471.

26 Prem Krishnaa, Girija Limaye, Meghana Nasre, and Prajakta Nimbhorkar. Envy-freeness and relaxed stability: Hardness and approximation algorithms. In Tobias Harks and Max Klimm, editors, *Algorithmic Game Theory - 13th International Symposium, SAGT 2020, Augsburg, Germany, September 16-18, 2020, Proceedings*, volume 12283 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2020. doi:10.1007/978-3-030-57980-7_13.

27 Krishnapriya A. M., Meghana Nasre, Prajakta Nimbhorkar, and Amit Rawat. How good are popular matchings? In Gianlorenzo D'Angelo, editor, *17th International Symposium on Experimental Algorithms, SEA 2018, June 27-29, 2018, L'Aquila, Italy*, volume 103 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SEA.2018.9.

28 David F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2 of *Series on Theoretical Computer Science*. WorldScientific, 2013. doi:10.1142/8591.

29 David F. Manlove, Robert W. Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Hard variants of stable marriage. *Theor. Comput. Sci.*, 276(1-2):261–279, 2002. doi:10.1016/S0304-3975(01)00206-7.

30 Alvin Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.

31 Alvin Roth. On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Econometrica*, 54(2):425–27, 1986.

32 Alvin E. Roth. The economics of matching: Stability and incentives. *Mathematics of Operations Research*, 7(4):617–628, 1982.

33 Yu Yokoi. Envy-free matchings with lower quotas. *Algorithmica*, 82(2):188–211, 2020. `doi:10.1007/s00453-018-0493-7`.

34 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007. `doi:10.4086/toc.2007.v003a006`.

A Examples

We give some examples that show the difficulty of implementing strategy-proof algorithms for HRT-MSLQ.

A.1 Incompatibility between Optimization and Strategy-proofness

Here, we provide two examples that show that solving HRT-MSLQ exactly is incompatible with strategy-proofness even if we ignore computational efficiency. This incompatibility holds even for restrictive models. The first example is an instance in the marriage model in which ties appear only in preference lists of hospitals. The second example is an instance in the uniform model in which ties appear only in preference lists of residents.

▶ **Example 15.** Consider the following instance I, consisting of two residents and three hospitals.

r_1:	h_1	h_2	h_3
r_2:	h_1	h_2	h_3

h_1 $[1,1]$:	$(r_1$	$r_2)$
h_2 $[1,1]$:	$(r_1$	$r_2)$
h_3 $[0,1]$:	$(r_1$	$r_2)$

Then, I has two stable matchings $M_1 = \{(r_1, h_1), (r_2, h_2)\}$ and $M_2 = \{(r_1, h_2), (r_2, h_1)\}$, both of which have a score of 3. Let A be an algorithm that outputs a stable matching with a maximum score for any instance of HRT-MSLQ. Without loss of generality, suppose that A returns M_1. Let I' be obtained from I by replacing r_2's list with "r_2 : h_1 h_3 h_2." Then, the stable matchings for I' are $M_3 = \{(r_1, h_1), (r_2, h_3)\}$ and $M_4 = \{(r_1, h_2), (r_2, h_1)\}$, which have scores 2 and 3, respectively. Since A should return one with a maximum score, the output is M_4, in which r_2 is assigned to h_1 while she is assigned to h_2 in M_1. As $h_1 \succ_{r_3} h_2$ in her true preference, this is a successful manipulation for r_2, and A is not strategy-proof.

Example 15 shows that there is no strategy-proof algorithm for HRT-MSLQ that attains an approximation factor better than 1.5 even if there are no computational constraints.

▶ **Example 16.** Consider the following instance I, consisting of six residents and five hospitals, where the notation "$\cdots$" at the tail of lists denotes an arbitrary strict order of all agents missing in the list.

r_1:	h_1	$\cdots$		
r_2:	h_3	h_2	h_1	$\cdots$
r_3:	h_3	$\cdots$		
r_4:	$(h_3$	$h_4)$	$\cdots$	
r_5:	h_4	$\cdots$		
r_6:	h_4	h_5	h_1	$\cdots$

h_1 $[1,2]$:	r_1	r_2	r_6	$\cdots$
h_2 $[1,2]$:	r_2	$\cdots$		
h_3 $[1,2]$:	r_3	r_4	r_2	$\cdots$
h_4 $[1,2]$:	r_5	r_4	r_6	$\cdots$
h_5 $[1,2]$:	r_6	$\cdots$		

This instance I has two stable matchings

$$M_1 = \{(r_1,h_1),(r_2,h_2),(r_3,h_3),(r_4,h_3),(r_5,h_4),(r_6,h_4)\}, \text{ and}$$
$$M_2 = \{(r_1,h_1),(r_2,h_3),(r_3,h_3),(r_4,h_4),(r_5,h_4),(r_6,h_5)\},$$

both of which have a score of 4. Let A be an algorithm that outputs an optimal solution for any input. Then, A must output either M_1 or M_2.

Suppose that A outputs M_1. Let I' be an instance obtained by replacing r_2's preference list from "r_2 : h_3 h_2 $h_1 \cdots$" to "r_2 : h_3 h_1 $h_2 \cdots$." Then, the stable matchings I' admits are M_2 and $M_1' = \{(r_1,h_1),(r_2,h_1),(r_3,h_3),(r_4,h_3),(r_5,h_4),(r_6,h_4)\}$, whose score is 3. Hence, A must output M_2. As a result, r_2 is assigned to a better hospital h_3 than h_2, so this manipulation is successful.

If A outputs M_2, then r_6 can successfully manipulate the result by changing her list from "r_6 : h_4 h_5 $h_1 \cdots$" to "r_6 : h_4 h_1 $h_5 \cdots$." The instance obtained by this manipulation has two stable matchings M_1 and $M_2' = \{(r_1,h_1),(r_2,h_3),(r_3,h_3),(r_4,h_4),(r_5,h_4),(r_6,h_1)\}$, whose score is 3. Hence, A must output M_1 and r_6 is assigned to h_4, which is better than h_5.

A.2 Absence of Strategy-proofness in Adaptive Tie-breaking

We provide an example that demonstrates that introducing a greedy tie-breaking method into the resident-oriented Gale–Shapley algorithm in an adaptive manner destroys the strategy-proofness for residents.

▶ **Example 17.** Consider the following instance I (in the uniform model), consisting of five residents and three hospitals.

r_1:	h_1	h_2	h_3
r_2:	$(h_1$	$h_2)$	h_3
r_3:	h_1	h_2	h_3
r_4:	h_2	h_1	h_3
r_5:	h_1	h_3	h_2

h_1 $[1,2]$:	r_2	r_3	r_5	r_1	r_4
h_2 $[1,2]$:	r_2	r_4	r_1	r_3	r_5
h_3 $[1,2]$:	r_1	r_2	r_3	r_4	r_5

Consider an algorithm that is basically the resident-oriented Gale–Shapley algorithm and let each resident prioritize deficient hospitals over sufficient hospitals among the hospitals in the same tie. Its one possible execution is as follows. First, r_1 proposes to h_1 and is accepted. Next, as h_1 is sufficient while h_2 is deficient, r_2 proposes to h_2 and is accepted. If we apply the ordinary Gale–Shapley procedure afterward, then we obtain a matching $\{(r_1,h_3),(r_2,h_2),(r_3,h_1),(r_4,h_2),(r_5,h_1)\}$. Thus, r_1 is assigned to her third choice.

Let I' be an instance obtained by swapping h_1 and h_2 in r_1's preference list. If we run the same algorithm for I', then r_1 first proposes to h_2. Next, as h_2 is sufficient while h_1 is deficient, r_2 proposes to h_1 and is accepted. By applying the ordinary Gale–Shapley procedure afterward, we obtain $\{(r_1,h_2),(r_2,h_1),(r_3,h_1),(r_4,h_2),(r_5,h_3)\}$. Thus, r_1 is assigned to a hospital h_2, which is her second choice in her original list. Therefore, this manipulation is successful for r_1.

Online Scheduling on Identical Machines with a Metric State Space

Hiromichi Goko
Toyota Motor Corporation, Aichi, Japan

Akitoshi Kawamura
Kyoto University, Japan

Yasushi Kawase
University of Tokyo, Japan

Kazuhisa Makino
Kyoto University, Japan

Hanna Sumita
Tokyo Institute of Technology, Japan

Abstract

This paper introduces an online scheduling problem on m identical machines with a metric state space, which generalizes the classical online scheduling problem on identical machines, the online traveling salesman problem, and the online dial-a-ride problem. Each job is associated with a source state, a destination state, a processing time, and a release time. Each machine can process a job on and after its release time. Before processing a job, a machine needs to change its state to the source state (in a time corresponding to the distance), and after the process of the job, the machine's state becomes the destination state. While related research deals with a model in which only release times are unknown to the algorithm, this paper focuses on a general model in which destination states and processing times are also unknown. The main result of this paper is to propose a $O(\log m/\log\log m)$-competitive online algorithm for the problem, which is best possible. A key approach is to divide the difficulty of the problem. To cope with unknown release times, we provide frameworks to produce a $\min\{2\rho+1/2,\ \rho+2\}$-competitive algorithm using a ρ-competitive algorithm for a basic case where all jobs are released at time 0. Then, focusing on unknown destination states and processing times, we construct an $O(\log m/\log\log m)$-competitive algorithm for the basic case. We also provide improved algorithms for some special cases.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online scheduling, Competitive analysis, Online dial-a-ride

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.32

Funding This work was partially supported by the joint project of Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society", JST PRESTO Grant Number JPMJPR2122, and JSPS KAKENHI Grant Numbers JP17K12646, JP19K22841, JP20H05967, JP20K19739, JP21K17708, and JP21H03397.

1 Introduction

For a metric space $\boldsymbol{M} = (X, d)$ and a positive integer m, we consider the following $(\boldsymbol{M}, m)$-*scheduling problem.* We have m identical machines, which work in parallel. Each machine i has a *state* $s_i(t)$ in X at time $t \in \mathbb{R}_+$, and moves along a path P in $\boldsymbol{M}$ to change a state x to a state y, where each machine is assumed to move in $\boldsymbol{M}$ with at most unit speed. All machines are initially (i.e., at time 0) located at the origin o in X and need to return to o at the end. Machines process jobs given in an online fashion. Each job j has two states called the *source* state $a_j \in X$ and *destination* state $b_j \in X$. It also has the processing time

$p_j \in \mathbb{R}_+$ and the release time $r_j \in \mathbb{R}_+$. Job j appears on r_j, can be processed on and after r_j, and requires p_j time to process it. To processes job j, machine i first needs to change its state to the source state a_j and reaches the destination state b_j afterward. We assume that $p_j \geq d(a_j, b_j)$, which means that $p_j - d(a_j, b_j)\,(\geq 0)$ time is additionally required to process the job j. The jobs are *non-preemptive*, i.e., once a machine starts to process a job, it must finish the processing of the job without doing anything else. In addition, any job has to be processed by exactly one machine. The $(\boldsymbol{M}, m)$-scheduling problem is to find a schedule to minimize the *makespan*, that is, the time when all machines return to the origin o after completing all jobs.

We study the *real-time online* version of this scheduling problem, called the *online* $(\boldsymbol{M}, m)$*-scheduling problem*. No information on jobs is available before their release time. Namely, we first see the job j at time r_j, and we do not know if it comes or not before time r_j. There exist two information models, called the *complete* and *incomplete* models, for the online $(\boldsymbol{M}, m)$-scheduling problem, where we mainly study the incomplete model. In the complete model, all the information of the job j are revealed at the release time r_j. On the other hand, in the incomplete model, the processing time p_j and destination state b_j are still unknown at time r_j and revealed when the job j is completed by some machine. Several fundamental cases of the online $(\boldsymbol{M}, m)$-scheduling problem in the complete information model have been studied in the literature of scheduling and combinatorial optimization [1–10, 17, 18], as described in the next subsection. On the other hand, little is known for the incomplete information model [19, 21]. However, there are a number of practical situations for which the incomplete model is suitable. For example, in the scheduling problems such as repairing companies, we do not know in advance the *exact* processing time for jobs as well as their situations (or states) when finishing them. Classical taxi-hailing services only collect the pick-up locations when customers phone the companies, and elevator systems are usually equipped with only landing call buttons, although the systems can get information on the directions (i.e., up and down) of the requests. Our goal is to design online algorithms for the online $(\boldsymbol{M}, m)$-scheduling problem under several natural settings of $\boldsymbol{M}$ and m. We also consider designing online algorithms for the *basic online* $(\boldsymbol{M}, m)$*-scheduling problem*, where all jobs are released at time 0, since it turns out that any algorithm for the basic problem can be extended to the one for the general problem. We analyze the performance of online algorithms by the *competitive ratio*, that is, the ratio between the optimal makespan and the one of the schedule obtained by the online algorithm.

1.1 Previous work

One of the simplest cases of the online $(\boldsymbol{M}, m)$-scheduling problem is the case when the metric space consists of a single point, i.e., the states of the machines are fixed. We denote the trivial metric as $\mathbb{R}^0$. For the online $(\mathbb{R}^0, m)$-scheduling problem, the following greedy algorithm is $(2 - 1/m)$-competitive, and this is best possible [13–15, 21]: assign an unprocessed job to any available machine anytime if possible. This result is regardless of information models. Shmoys et al. [21] focus on the incomplete information model and introduced a technique to convert a ρ-competitive algorithm for the basic online $(\mathbb{R}^0, m)$-scheduling problem into a 2ρ-competitive one for the general online $(\mathbb{R}^0, m)$-scheduling problem.

An important special case of the problem is when the processing of each job does not change the state, i.e., $a_j = b_j$ for all jobs j. Such a situation appears in a production system with sequence-dependent setup or changeover, e.g., mold setup, die setup, or color setup [16, 20]. To process a job in a plastic production system, we must attach the corresponding injection mold to an injection machine. Thus, we need setup time before and after processing the job.

Gambosi and Nicosia [12] studied an online version (in the one-by-one model) of scheduling with setup costs. Furthermore, a particular case where the processing time of every job equals 0 is studied as the online traveling salesman problem (TSP) [2, 3, 7, 8, 17]. The competitive ratio of the online TSP is 2 for any metric and any number of machines [2,17]. In addition, the competitive ratio of the single-server ($m = 1$) online TSP with the line metric is $(9 + \sqrt{17})/8 \approx 1.64$ [2, 3, 7]. We remark that the online TSP is included in the complete information model.

When the processing time of each job is equal to its travel distance, i.e., $p_j = d(a_j, b_j)$ for all jobs j, our problem is studied under the name of the *online dial-a-ride problem*. There is a large body of studies on the online dial-a-ride problem [1, 4–6, 9, 10, 18, 19], but only Lipmann et al. [19] addresses the incomplete information model. Lipmann et al. [19] showed that upper and lower bounds of the single machine online dial-a-ride problem are 4 and $1 + \frac{3}{2}\sqrt{2}$ (≈ 3.121), respectively. For the upper bound, they designed an algorithm, called BOUNCER, which decides a process order based on a solution to the online TSP over the source states. In the algorithm, every time the machine completes a job, the machine changes the state to the source state of the job. Thus, the makespan is the length of the TSP tour plus twice the sum of processing times. Because a 2-competitive algorithm is known for the online TSP, the BOUNCER algorithm is 4-competitive. Note that the BOUNCER algorithm is easily extended to the online $(\boldsymbol{M}, m)$-scheduling problem, but its competitive ratio is $2 + 2m$. We also note that in their paper, the processing time and destination state of a job are revealed at the moment when some machine starts processing the job. When $m = 1$ and preemption is not allowed, this is the same as our model. However, our model is more general when $m > 1$. For the online dial-a-ride problem under the complete information model, Ascheuer et al. [1] analyzed two natural algorithms, which they call IGNORE and REPLAN, for the single machine setting. They also provide an algorithm called SMARTSTART, which is 2-competitive for any metric and any number of machines. This is best possible since a lower bound of the competitive ratio is 2 even for the online TSP [2].

1.2 Our results

Our main result is to provide an $O(\log m/ \log\log m)$-competitive algorithm (Theorem 8) and prove that this is best possible up to a constant factor (Theorem 7) among algorithms for the general $(\boldsymbol{M}, m)$-scheduling problem. We summarize our results and existing ones in Table 1.

We describe the techniques to obtain our results. Our approach is to divide the difficulty of our problem into two types of unknown information; one is a release time (online arrival), and the other consists of a destination state and a processing time.

First, to cope with the unknown release times, we show a framework to design an algorithm using one for the basic online $(\boldsymbol{M}, m)$-scheduling problem. We describe this in Section 3. Roughly speaking, our framework computes a schedule by repeatedly applying the basic case algorithm. As rules for applying the algorithm, we adopt three natural strategies, called IGNORE, REPLAN, and SMARTSTART, proposed for the online dial-a-ride problem in the complete information model [1]. We analyze these strategies in detail using three factors that contribute to the makespan. The analysis implies that a ρ-competitive algorithm for the basic online $(\boldsymbol{M}, m)$-scheduling problem can be converted into a $\min\{2\rho + 1/2,\ \rho + 2\}$-competitive algorithm for the general case (Corollaries 2 and 4). We remark that this extends the result by Shmoys et al. [21]. They deal with only the fixed state case (i.e., $\boldsymbol{M} = \mathbb{R}^0$), and their method results in a 2ρ-competitive algorithm. If ρ is greater than 2, our method leads to a better competitive ratio. Consequently, we only need to focus on the basic case.

Then, focusing only on unknown destination states and processing times, we present an $O(\log m/\log\log m)$-competitive algorithm for the basic online $(\boldsymbol{M}, m)$-scheduling problem (Algorithm 1) in Section 4.1. Our algorithm first partition the jobs into m groups and assigns one group to each machine based only on the information of source states. To shorten the makespan, we set machines that complete their assigned groups to process a group that has not yet complete. It is not a good idea to assign additional machines to an uncompleted group immediately and greedily: this method may cause a machine making a vain effort to assist one completing soon. Thus, we control reassignments. Namely, we appropriately wait and reassign jobs so that the number of assisting machines increases exponentially with base q where $q^q > m \geq (q-1)^{q-1}$. Our analysis is not only elementary but also it shows a tight competitive ratio for the basic problem. This result together with the above conversion implies our main algorithmic result. We can see the ratio is the best possible because we show that any online algorithm is $\Omega(\log m/\log\log m)$-competitive for the basic online $(\boldsymbol{M}, m)$-scheduling problem. Note that this lower bound also holds for the cases of (i) $p_j = d(a_j, b_j)$ for all job j, (ii) $a_j = b_j$ for all job j, and (iii) the destination state and the processing time of each job are revealed at the moment when some machine starts processing the job.

We also discuss the competitive ratio of special cases in Section 5. One is a single machine case. We show that Algorithm 1 is 3-competitive for the basic online $(\boldsymbol{M}, 1)$-scheduling problem, and prove that no online algorithm has the competitive ratio better than 2.255 and 3.181 for the basic and the general problems, respectively. We note that our bound of 3.181 improves the best known lower bound of $1 + \frac{3\sqrt{2}}{2} \approx 3.12$ for the online dial-a-ride problem under the incomplete information model [19]. Another is a two-machine case. We improve Algorithm 1 to obtain a 3.5-competitive and a $(4 + \frac{\sqrt{3}}{2})$-competitive algorithm for the basic and the general online $(\boldsymbol{M}, 2)$-scheduling problem, respectively. In addition, when the optimal values of the TSP with a single machine and m machines are close, we provide a 13/3-competitive algorithm for the basic online $(\mathbb{R}, m)$-scheduling problem.

Finally, we discuss several other variants of our problem in Section 6: minimizing the total completion time is hopeless, the open setting (the machines do not need to return to the origin) can easily be reduced to the closed setting (the machines do not need to return to the origin) with loss of factor 2, and preemption does not help if the destination state and the processing time of each job are revealed upon completion.

Due to space limitations, some proofs are deferred to Appendix A.

Table 1 Summary of our results and previous work.

		incomplete info. model		complete info. model	
release time	# machines	upper bound	lower bound	upper bound	lower bound
general	m	$\boldsymbol{\Theta}(\frac{\log m}{\log\log m})$ (Thms. 8, 7)		2 [1]	
	1	4 (Thm. 13)	**3.181** (Thm. 15)		
0 (basic)	m	$\boldsymbol{\Theta}(\frac{\log m}{\log\log m})$ (Thms. 9, 7)		1	
	1	**3** (Thm. 12)	**2.255** (Thm. 14)		

2 Preliminaries

An instance of the online $(\boldsymbol{M}, m)$-scheduling problem is specified by a metric space $\boldsymbol{M} = (X, d)$ with a distinguished origin $o \in X$, the number of machines m, and a set of jobs J. Let $[m] = \{1, 2, \ldots, m\}$ be the set of machines. Note that $d \colon X \times X \to \mathbb{R}_+$ is a function such that for any $x, y, z \in X$, the following holds: (i) $d(x, y) = 0 \iff x = y$, (ii) $d(x, y) = d(y, x)$, and (iii) $d(x, z) \leq d(x, y) + d(y, z)$. We assume that $\boldsymbol{M}$ is path-connected, i.e., for any pair of points $x, y \in X$, there is a continuous path $\gamma\colon [0, 1] \to X$ with $\gamma(0) = x$ and $\gamma(1) = y$ of length $d(x, y)$. Examples of such metric spaces are the real line $\mathbb{R}$, the Euclidean plane $\mathbb{R}^2$, and a circle $\mathbb{R}/\mathbb{Z}$, where we assume that each set has the Euclidean distance. Each job $j \in J$ is associated with a tuple $(a_j, b_j, p_j, r_j) \in X \times X \times \mathbb{R}_+ \times \mathbb{R}_+$, where a_j and b_j are respectively the source and the destination states, p_j is the processing time, and r_j is the release time of job j. When processing a job j by a machine in state s, it is first necessary to change the state of machine to a_j in time $d(s, a_j)$. Then after the machine starts to process job j, the machine's state is changed continuously from a_j, and finally reaches b_j. Possibly the process is done without state change (e.g., injection molds remain unchanged during production of one item). A job j is called *empty* if $a_j = b_j$ and $p_j = 0$. Assume that the state of each machine i is the origin o at time 0 (i.e., $s_i(0) = o$), the state can be changed in at most unit speed (i.e., $d(s_i(t), s_i(t')) \leq |t - t'|$ for all $i \in [m]$ and $t, t' \in \mathbb{R}_+$), and the processing time p_j is at least $d(a_j, b_j)$ for all job j. Each machine can process at most one job at a time. We do not allow preemption; once a machine starts processing a job, it is not permitted to stop until the process is completed. The objective of the problem is to minimize the completion time (makespan), which is the time when the machines have completed all jobs and returned to o.

A (feasible) *schedule* is a sequence of processing and states change of the machines satisfying the following: (1) the state of each machine is the origin o at time 0 and at the end, (2) the state of each machine is changed in at most unit speed, and (3) every job is processed on or after its release time. An online algorithm decides a partial schedule in real-time. We focus on the *incomplete* information model in which an online algorithm has no information about the destination state and the processing time of each job until the job is completed. The source state of each job is revealed at its release time.

We evaluate the performance of an online algorithm by the *competitive ratio*. Throughout the paper, we only care about deterministic online algorithms. For a problem instance I, we denote the completion time by an algorithm ALG and an optimal offline algorithm OPT by ALG(I) and OPT(I), respectively. We assume that the optimal offline algorithm knows the information of all the jobs in advance. An algorithm ALG is said to be c-competitive if $\text{ALG}(I) \leq c \cdot \text{OPT}(I)$ for any instance I of the online (basic) $(\boldsymbol{M}, m)$-scheduling problem. We refer to the schedule of the optimal offline algorithm as the optimal offline schedule.

3 Reduction to the Basic Problem

In this section, we consider three natural strategies, called IGNORE, REPLAN, and SMARTSTART, to convert an algorithm for the basic online $(\boldsymbol{M}, m)$-scheduling problem into an algorithm for the online $(\boldsymbol{M}, m)$-scheduling problem. These strategies are applied to the (complete information) online dial-a-ride problem in [1]. IGNORE is a strategy that runs an optimal subsequent schedule (which can be computed in the complete information setting) for unprocessed jobs if they exist, ignoring all new jobs until all machines finish the assigned jobs and come back to the origin o. REPLAN is a greedy strategy that stops the current plan whenever a new job is released, makes all the machines return to the origin o right after the current process, and starts a replanned optimal subsequent schedule for the remaining jobs. We remark that these two strategies decide when to start independently of processing times, while SMARTSTART uses the information. SMARTSTART is similar to IGNORE, but it may keep the machines idle for a time depending on an estimated processing time of remaining jobs (which is, for example, calculated by assuming that the processing time is all zero). These strategies do not move machines until some jobs are released, and hence we do not need to consider the instance with no jobs, called the empty instance, in the competitive analysis.

Let ALG_0 be an algorithm for the basic online $(\boldsymbol{M}, m)$-scheduling problem, and let I_0 be an instance. We analyze our conversions by the following three factors that contribute to the makespan: (i) the optimal completion time $\mathrm{OPT}(I_0)$, (ii) the mean sum $p(I_0)/m$ of processing times per machine, where $p(I_0) = \sum_{j \in J} p_j$ for the jobs in I_0, and (iii) the lower bound $\mathrm{LB}(I_0)$ on the minimum completion time for I_0, which is given by the optimal value of the *m-traveling salesman problem* (m-TSP) over the source states. We refer to the m-TSP as the problem of finding m tours that visit every location so as to minimize the length of the longest tour [11]. The 1-TSP coincides with TSP. In what follows, we assume that

$$\mathrm{ALG}_0(I_0) \le \alpha \mathrm{OPT}(I_0) + \beta \cdot p(I_0)/m + \gamma \mathrm{LB}(I_0), \tag{1}$$

where α, β, and γ are nonnegative reals independent from I_0. Note that ALG_0 is $(\alpha+\beta+\gamma)$-competitive because $\mathrm{OPT}(I_0) \ge p(I_0)/m$ and $\mathrm{OPT}(I_0) \ge \mathrm{LB}(I_0)$. In addition, $\alpha + \gamma \ge 1$ because $\mathrm{ALG}_0(I_0) \le (\alpha + \gamma) \cdot \mathrm{OPT}(I_0)$ for any instance I_0 with $p(I_0) = 0$. For a set S' of jobs, we denote by S'_0 the instance of the basic $(\boldsymbol{M}, m)$-scheduling problem with job set S'.

We first consider the following IGNORE algorithm. The machines remain idle (i.e., the machines are in the origin and not working) until a non-empty set S of unprocessed jobs appear. The algorithm then immediately processes S following the schedule obtained by ALG_0 for S_0. We refer to this schedule as a subschedule for S. All jobs that arrive during the execution of the subschedule are temporarily ignored until the subschedule is completed and all machines become idle again. Then, the algorithm continues the same process.

▶ **Theorem 1.** *IGNORE is $(2\alpha+\beta+2\gamma+1/2)$-competitive for the online $(\boldsymbol{M}, m)$-scheduling problem.*

Proof. We fix a non-empty instance I. Let t^* be the last released time of jobs in I.

Suppose that the machines are idle at time t^*. Let R be the set of jobs processed in the last subschedule of the IGNORE algorithm. By construction, the length of the last subschedule is $\mathrm{ALG}_0(R_0) \le \alpha\mathrm{OPT}(R_0) + \beta p(R_0)/m + \gamma\mathrm{LB}(R_0) \le (\alpha+\beta+\gamma)\mathrm{OPT}(I)$. Since $t^* \le \mathrm{OPT}(I)$ and $\alpha + \beta \ge 1$, it holds that

$$\begin{aligned}\mathrm{IGNORE}(I) = t^* + \mathrm{ALG}_0(R_0) &\le t^* + (\alpha+\beta+\gamma)\cdot\mathrm{OPT}(I)\\ &\le (1+\alpha+\beta+\gamma)\mathrm{OPT}(I) \le (2\alpha+\beta+2\gamma+1/2)\mathrm{OPT}(I).\end{aligned}$$

Now suppose that some machines are not idle at time t^*. Then t^* is in the second last subschedule of the algorithm, and the last subschedule starts right after the second last subschedule ends. Let R and S be the sets of jobs processed in the last and the second last subschedule, respectively. We denote by $r' = \min_{(a,b,p,r)\in R} r$.

Let q_i be the state of machine i at time r' in the optimal offline schedule for I. Thus, $\max_{i\in[m]} d(o, q_i) \le r' \le \mathrm{OPT}(I)$ and $\mathrm{OPT}(I) \ge 2\max_{i\in[m]} d(o, q_i)$. For the instance R_0, the minimum makespan is $\mathrm{OPT}(R_0)$. On the other hand, when we first move each machine i to state q_i, and then imitate an optimal offline schedule for I after time r', ignoring jobs not in R, the makespan will be $\max_{i\in[m]} d(o, q_i) + \mathrm{OPT}(I) - r'$. Thus, we see that $\mathrm{OPT}(R_0) \le \max_{i\in[m]} d(o, q_i) + \mathrm{OPT}(I) - r'$. Hence, we obtain that

$$
\begin{aligned}
&\mathrm{IGNORE}(I)\\
&\le r' + \mathrm{ALG}_0(S_0) + \mathrm{ALG}_0(R_0)\\
&\le r' + (\alpha+\gamma)\mathrm{OPT}(S_0) + (\alpha+\gamma)\mathrm{OPT}(R_0) + \beta p(S_0)/m + \beta p(R_0)/m\\
&\le r' + (\alpha+\gamma)\mathrm{OPT}(I) + (\alpha+\gamma)\mathrm{OPT}(R_0) + \beta p(I_0)/m + (\alpha+\gamma-1)(r' - \max_i d(o,q_i))\\
&\le (\alpha+\gamma)\left(r' + \mathrm{OPT}(R_0) - \max_{i\in[m]} d(o,q_i)\right) + (\alpha+\beta+\gamma)\mathrm{OPT}(I) + \max_{i\in[m]} d(o,q_i)\\
&\le (2\alpha+\beta+2\gamma+1/2)\mathrm{OPT}(I). \qquad ◀
\end{aligned}
$$

▶ **Corollary 2.** *If there exists a ρ-competitive algorithm for the basic online $(\boldsymbol{M}, m)$-scheduling problem, then there exists a $(2\rho+1/2)$-competitive algorithm for the online $(\boldsymbol{M}, m)$-scheduling problem.*

Next, we consider the following REPLAN algorithm. When a new job is released, the REPLAN algorithm calls all the machines, and each machine comes back to the origin o immediately after completing the currently processing job. After all machines are returned, it processes the unprocessed jobs S following the schedule obtained by ALG_0 for S_0.

▶ **Theorem 3.** *REPLAN is $(\alpha+\beta+\gamma+2)$-competitive for the online $(\boldsymbol{M}, m)$-scheduling problem.*

Proof. We fix a non-empty instance I. Let t^* be the last release time of jobs appearing in I, and let R be the set of unprocessed jobs at time t^*.

At the time t^*, all machines are made to directly return to the origin o as soon as possible. A machine that is not processing any job (including on the way to some job) at time t^* can return in t^* $(\le \mathrm{OPT}(I))$ time, and a machine that is processing job j can return in $p_j + d(b_j, o)$ $(\le \mathrm{OPT}(I))$ time. Thus, all machines return in at most $\mathrm{OPT}(I)$ time. After all the machines have returned to the origin o, it takes $\mathrm{ALG}_0(R_0) \le \alpha\mathrm{OPT}(R_0) + \beta p(R_0)/m + \gamma\mathrm{LB}(R_0) \le (\alpha+\beta+\gamma)\mathrm{OPT}(I)$ time to complete all the jobs in R. Hence, the completion time by REPLAN is at most

$$\mathrm{REPLAN}(I) \le t^* + \mathrm{OPT}(I) + (\alpha+\beta+\gamma)\mathrm{OPT}(I) \le (\alpha+\beta+\gamma+2)\mathrm{OPT}(I). \qquad ◀$$

▶ **Corollary 4.** *If there exists a ρ-competitive algorithm for the basic online $(\boldsymbol{M}, m)$-scheduling problem, then there exists a $(\rho+2)$-competitive algorithm for the online $(\boldsymbol{M}, m)$-scheduling problem.*

Finally, we discuss the SMARTSTART strategy, which is originally proposed for the complete information model [1]. SMARTSTART calculates the minimum completion time T for the unprocessed jobs and waits until the time T. However, this is impossible in the incomplete information model. Therefore, we use $\mathrm{LB}(I_0)$ as an alternative parameter to decide when to start.

To be precise, the algorithm SMARTSTART has a fixed parameter $\theta \ge 0$, which will be optimized later. When the machines are idle, and there is a non-empty set S of unprocessed jobs, the algorithm computes $\mathrm{LB}(S_0)$. The algorithm keeps machines idle until the time $\theta \cdot \mathrm{LB}(S_0)$ if it is before the time. Then, it immediately processes S following the schedule obtained by ALG_0 for S_0. The algorithm ignores all jobs that arrived during the execution of the subschedule until the subschedule is completed and all machines become idle again.

▶ **Theorem 5.** *SMARTSTART is* $\frac{6\alpha+4\beta+4\gamma+1+\sqrt{(2\alpha+1)^2+8\gamma}}{4}$*-competitive for the online* $(\boldsymbol{M}, m)$*-scheduling problem by setting* $\theta = \frac{2\alpha+1+\sqrt{(2\alpha+1)^2+8\gamma}}{4}$.

We remark that similar statements to Theorems 1, 3, and 5 hold for other settings such as complete information and preemptive setting. Corollaries 2 and 4 yield that we only need to construct an $O(\log m/\log\log m)$-competitive algorithm for the basic case, and we do this in the next section. We mention that SMARTSTART derives better competitive algorithms than others for the single and two machine cases as shown Section 5.

4 Algorithm and Hardness of the Basic Problem

In this section, we show our main results for the basic online $(\boldsymbol{M}, m)$-scheduling problem. We first present an $O(\log m/\log\log m)$-competitive algorithm in Section 4.1, and then prove that this is best possible among deterministic online algorithms in Section 4.2.

Specifically, we show the following theorems.

▶ **Theorem 6.** *For any metric* $\boldsymbol{M}$*, there exists an* $O(\log m/\log\log m)$*-competitive algorithm for the basic online* $(\boldsymbol{M}, m)$*-scheduling problem.*

▶ **Theorem 7.** *There exists a metric* $\boldsymbol{M}$ *such that every online algorithm is* $\Omega(\log m/\log\log m)$*-competitive for the basic online* $(\boldsymbol{M}, m)$*-scheduling problem.*

We remark that Theorem 6 together with Theorem 1 or 3 in the previous section implies our main result for the general case.

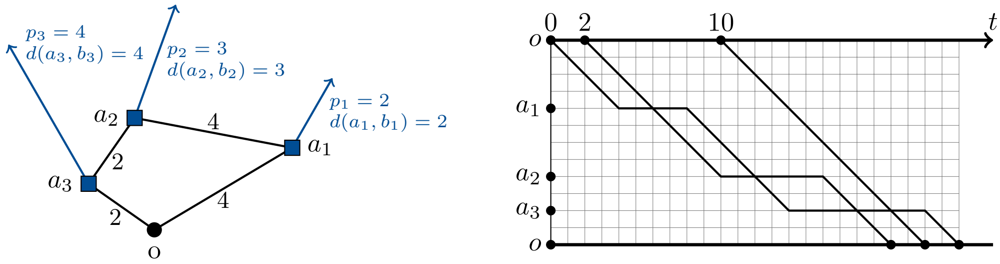
▶ **Theorem 8.** *For any metric* $\boldsymbol{M}$*, there exists an* $O(\log m/\log\log m)$*-competitive algorithm for the online* $(\boldsymbol{M}, m)$*-scheduling problem.*

The lower bound on the basic case also applies to the general case, and hence we see that the competitive ratio of $O(\log m/\log\log m)$ is best possible.

4.1 Upper bound

Let I be an instance of the basic online $(\boldsymbol{M}, m)$-scheduling problem. Recall that we know the source states of all the jobs. A simple way to construct a reasonable schedule for the problem would be to use a shortest tour for the source states. More precisely, we solve the m-TSP over the source states. Let $C_1, \ldots, C_m$ be the optimal solution of the m-TSP over the source states. Then each machine i processes jobs appearing in (directed) tour C_i in the order of C_i. After processing a job, the machine returns to its source state and moves on to the next job. Unfortunately, the competitive ratio of this simple method is $\Omega(m)$ even when the metric is $\mathbb{R}^2$ (see Example 21 in Appendix).

The reason why the above simple method did not work well is that some machines spend time in idleness. We resolve this issue by carefully reassigning idle machines to an *uncompleted* tour. A tour is said to be completed if all the jobs appearing in the tour have been completed, and all the machines assigned to the tour have returned to the origin o. We describe the idea of our algorithm. We first assign machine i to each tour C_i, and each machine processes jobs in the assigned tour in the same way as the simple method. If a tour takes a long time to be completed, our algorithm additionally assigns idle machines to the tour. Each machine travels along the assigned tour. When a machine arrives at the source of an unprocessed job, then it processes the job, returns to its source, and continues the travel (see Figure 1). The point of our algorithm is that it does not reassign machines immediately and greedily. Our algorithm reassigns machines so that the number of machines assigned to each uncompleted tour increases exponentially by the following rule. To avoid reassigning to a tour that will be completed soon, the algorithm waits for a certain number of idle machines before reassigning. This exponential increase in the number of assigned machines will help us analyze the competitive ratio of the algorithm.



(a) A tour C_i.

(b) Diagram of machines' locations.

Figure 1 A process of a tour when three machines assigned at time 0, 2, and 10, respectively.

Let q be a positive integer, which will be set later. The execution of our algorithm consists of phases. In phase k, our algorithm adds idle machines to each uncompleted tour so that the number of assigned machines for the tour is q^{k-1}. The phase k continues until the number of uncompleted tours decreases to $\lfloor m/q^k \rfloor$. The algorithm terminates in phase k^* with $m/q^{k^*-1} \geq 1 > m/q^{k^*}$, that is, $k^* = \lfloor \log_q m \rfloor + 1$ $(\leq q)$. Then, we set q so as to bound the makespan. Intuitively, the number of phases increases as q becomes smaller, and the time length of each phase increases as q becomes larger. We set q to the integer such that $q^q > m \geq (q-1)^{q-1}$, taking the tradeoff between them into account. Note that $q = \Theta(\log m/ \log\log m)$.

Our algorithm is summarized as Algorithm 1. Note that, at line 6, the number of uncompleted tours may become less than $\lfloor m/q^k \rfloor$ because of multiple tours being completed at the same time. In such a case, we break ties arbitrarily to choose $\lfloor m/q^k \rfloor$ tours for phase k and carry the other (already completed) tours over into the next phase.

Algorithm 1 The proposed algorithm for the basic online$(\boldsymbol{M}, m)$-scheduling problem.

```
1 Let $q$ be the integer such that $q^q > m \geq (q-1)^{q-1}$;
2 Set $k^* \leftarrow \lfloor \log_q m \rfloor + 1$;
3 Compute optimal $m$-TSP tours $C_1, \ldots, C_m$ over the sources;
4 for $k \leftarrow 1, 2, \ldots, k^*$ do
5     Assign available machines to the uncompleted tours so that the number of
        assigned machines becomes $q^{k-1}$ for each tour;
6     Continue the process for each uncompleted tour until the number of uncompleted
        tours becomes $\lfloor m/q^k \rfloor$;
```

We prove that Algorithm 1 is $O(\log m/ \log\log m)$-competitive for any metric $\boldsymbol{M}$, which immediately yields Theorem 6.

▶ **Theorem 9.** *For any metric $\boldsymbol{M}$, Algorithm 1 is $O(\log m/ \log\log m)$-competitive for the basic online $(\boldsymbol{M}, m)$-scheduling problem.*

We prove this theorem in the following. We define ℓ_i to be the length of tour C_i (over the sources) and J_i to be the set of all jobs corresponding to C_i for each $i \in [m]$. Let $h_i^{\mathrm{sum}} = \sum_{j \in J_i}(p_j + d(b_j, a_j))$ and $h_i^{\max} = \max_{j \in J_i}(p_j + d(b_j, a_j))$. Note that if one machine processes jobs in J_i according to C_i, then it takes at most $\ell_i + h_i^{\mathrm{sum}}$ time.

We first provide lower bounds on the optimal offline makespan $\mathrm{OPT}(I)$.

▶ **Lemma 10.** $\mathrm{OPT}(I) \geq \max\{\max_{i\in[m]} \ell_i,\ \max_{i\in[m]} h_i^{\max},\ \sum_{i\in[m]} h_i^{\mathrm{sum}}/(2m)\}$.

Proof. As the optimal offline algorithm must reach all source states and $\max_{i\in[m]} \ell_i$ is the optimal value of m-TSP over the sources, we have $\mathrm{OPT}(I) \geq \max_{i\in[m]} \ell_i$.

Let j^* be a job such that $p_{j^*} + d(b_{j^*}, a_{j^*}) = \max_{i\in[m]} h_i^{\max}$ $(= \max_{j\in J}(p_j + d(b_j, a_j)))$. Since j^* must be processed at some time, we have

$$\mathrm{OPT}(I) \geq d(o, a_{j^*}) + p^* + d(b_{j^*}, o) \geq p^* + d(b_{j^*}, a_{j^*}) = \max_{i\in[m]} h_i^{\max},$$

where the second inequality holds by the triangle inequality.

Finally, the makespan is not less than the mean sum of processing times spent by each machine. By this and the assumption that $p \geq d(a, b)$ for each job, we obtain

$$\mathrm{OPT}(I) \geq \frac{1}{m}\sum_{j\in J} p_j \geq \sum_{j\in J} \frac{p_j + d(a_j, b_j)}{2m} = \sum_{i\in[m]} \frac{h_i^{\mathrm{sum}}}{2m}. \qquad ◀$$

We analyze the completion time for each set of jobs J_i in terms of ℓ_i, $h_i^{\max}$, and h_i^{sum}.

▶ **Lemma 11.** *For a set of jobs J_i, let κ be the number of machines assigned for J_i throughout the algorithm. Then, the process for J_i is completed within $\ell_i + h_i^{\max} + \frac{h_i^{\mathrm{sum}}}{\kappa}$ from the time when the last machine was added.*

Proof. Throughout this proof, we only focus on the elapsed time after the last investment of machines for J_i. Let x and y be the machines that return to the origin o first and last, respectively. Suppose to the contrary that y returns to the origin o after time $\ell_i + h_i^{\max} + h_i^{\mathrm{sum}}/\kappa$. As the sum of the completion times of κ machines for J_i is at most $\kappa \cdot \ell_i + h_i^{\mathrm{sum}}$, x will complete the process on and before time $\ell_i + h_i^{\mathrm{sum}}/\kappa$. Thus, x returns to o at least $h_i^{\max}$ time earlier than y.

Consider the time when y starts processing the last job j^* in J_i. Here, y processes at least one job since otherwise y returns to the origin o by time ℓ_i, which is a contradiction. When working on j^*, the machine y must precede (or be at the same point as) the machine x, since otherwise y does not process j^*. Therefore, the time at which x return to the origin o is at most $p_{j^*} + d(b_{j^*}, a_{j^*})$ time earlier than y. However, this contradicts the assumption because $p_{j^*} + d(b_{j^*}, a_{j^*}) \leq h_i^{\max}$. ◀

Combining the above two lemmas, we can prove Theorem 9. For each phase $k \in [k^*]$, let $\tau_k \in \mathbb{R}_+$ be the time length of phase k, and let $S_k \subseteq [m]$ be the set of tours completed in phase k. Note that $|S_k| = \lfloor m/q^{k-1} \rfloor - \lfloor m/q^k \rfloor$ $(\forall k \in [k^*])$ and the makespan of the schedule obtained by the algorithm is $\sum_{k=1}^{k^*} \tau_k$. We will bound the value τ_k by using Lemma 11. Using the lemma in an intuitive way, we can obtain that $\tau_k \leq \max_{i\in S_k}\left(\ell_i + h_i^{\max} + \frac{h_i^{\mathrm{sum}}}{q^{k-1}}\right)$. However, this does not work well because it is far from $\sum_{i\in[m]} h_i^{\mathrm{sum}}/(2m)$, which is the only tool we have now to bound h_i^{sum} by $\mathrm{OPT}(I)$. Our main idea to overcome this issue is to use S_{k+1} instead of S_k, which allows us to evaluate τ_k as an average rather than a maximum. By combining this with the exponential increase in the number of machines in each phase, we can obtain the desired upper bound.

Proof of Theorem 9. It is sufficient to prove $\sum_{k=1}^{k^*} \tau_k = O(q) \cdot \mathrm{OPT}(I)$ because $O(q) = O(\log m / \log\log m)$.

We first bound τ_k for each phase $k \in [k^* - 1]$. By Lemma 11, we have $\tau_k \le \ell_i + h_i^{\max} + \frac{h_i^{\mathrm{sum}}}{q^{k-1}}$ for any $i \in S_{k+1}$. Hence, by Lemma 10, we obtain

$$\begin{aligned}\tau_k &\le \min_{i \in S_{k+1}} \left(\ell_i + h_i^{\max} + \frac{h_i^{\mathrm{sum}}}{q^{k-1}} \right) \le \frac{1}{|S_{k+1}|} \sum_{i \in S_{k+1}} \left(\ell_i + h_i^{\max} + \frac{h_i^{\mathrm{sum}}}{q^{k-1}} \right) \\ &\le 2\mathrm{OPT}(I) + \sum_{i \in S_{k+1}} \frac{h_i^{\mathrm{sum}}}{q^{k-1} \cdot |S_{k+1}|} = 2\mathrm{OPT}(I) + \frac{2m}{q^{k-1} \cdot |S_{k+1}|} \cdot \sum_{i \in S_{k+1}} \frac{h_i^{\mathrm{sum}}}{2m}.\end{aligned}$$

As $|S_{k+1}| = \lfloor m/q^k \rfloor - \lfloor m/q^{k+1} \rfloor \ge \lfloor m/q^k \rfloor - \frac{1}{q} \lfloor m/q^k \rfloor = (1 - 1/q) \lfloor m/q^k \rfloor$, we get

$$\frac{2m}{q^{k-1} \cdot |S_{k+1}|} \le 2q \cdot \frac{m/q^k}{\lfloor m/q^k \rfloor - \frac{1}{q} \lfloor m/q^k \rfloor} = 2q \cdot \frac{1}{1 - 1/q} \cdot \frac{m/q^k}{\lfloor m/q^k \rfloor} \le 8q,$$

where the second inequality holds by $q \ge 2$ and $m/q^k \ge 1$. Thus, we obtain

$$\tau_k \le 2\mathrm{OPT}(I) + 8q \sum_{i \in S_{k+1}} \frac{h_i^{\mathrm{sum}}}{2m}. \tag{2}$$

Next, we bound τ_{k^*}. By Lemmas 10 and 11, we have

$$\begin{aligned}\tau_{k^*} &\le \max_{i \in S_{k^*}} \left(\ell_i + h_i^{\max} + \frac{h_i^{\mathrm{sum}}}{q^{k^*-1}} \right) \le 2\mathrm{OPT}(I) + \max_{i \in S_{k^*}} \frac{h_i^{\mathrm{sum}}}{q^{k^*-1}} \\ &< 2\mathrm{OPT}(I) + 2q \cdot \max_{i \in S_{k^*}} \frac{h_i^{\mathrm{sum}}}{2m} \le (2 + 2q)\mathrm{OPT}(I), \end{aligned} \tag{3}$$

where the third inequality holds by $m < q^{k^*}$.

Hence, by (2) and (3), we obtain

$$\begin{aligned}\sum_{k=1}^{k^*} \tau_k &\le 2(k^* - 1) \cdot \mathrm{OPT}(I) + 8q \sum_{k=1}^{k^*-1} \sum_{i \in S_{k+1}} \frac{h_i^{\mathrm{sum}}}{2m} + (2 + 2q) \cdot \mathrm{OPT}(I) \\ &\le 4q \cdot \mathrm{OPT}(I) + 8q \cdot \sum_{i \in [m]} \frac{h_i^{\mathrm{sum}}}{2m} \le 4q \cdot \mathrm{OPT}(I) + 8q \cdot \mathrm{OPT}(I) = 12q \cdot \mathrm{OPT}(I).\end{aligned}$$

Therefore, Algorithm 1 is $O(\log m / \log\log m)$-competitive. ◀

Before concluding this subsection, we would like to mention that we can also construct a polynomial-time $O(\log m / \log\log m)$-competitive algorithm for the online $(\boldsymbol{M}, m)$-scheduling problem by using a constant approximation solution of m-TSP (which can be computed in polynomial-time [11]) instead of the optimal one in Algorithm 1.

4.2 Lower bound

We prove Theorem 7 that the competitive ratio $O(\log m / \log\log m)$ is best possible.

Proof of Theorem 7. We define a star-shaped metric $\boldsymbol{M} = (X, d)$ with $o = (0, 0)$ as follows:

$$X = \big(\mathbb{N} \times (0, 1]\big) \cup \{o\} \text{ and } d\big((i, x), (j, y)\big) = \begin{cases} |x - y| & \text{if } i = j, \\ x + y & \text{if } i \ne j. \end{cases}$$

Let us denote the end point $(i, 1) \in X$ by σ_i.

Let q be the positive integer such that $q^q \leq m < (q+1)^{q+1}$. For each $i \in [q^q]$, we prepare sufficiently many jobs (say m^2) that are either $(\sigma_i, \sigma_i, 0)$ or $(\sigma_i, \sigma_i, 1)$, depending on the actions taken by the online algorithm. We will refer to jobs $(\sigma_i, \sigma_i, 0)$ and $(\sigma_i, \sigma_i, 1)$ as empty and non-empty, respectively.

We fix an online algorithm. Then the adversary partitions the index set $[q^q]$ of end points into $q+1$ sets $S_1, S_2, \dots, S_{q+1}$ based on the behavior of the algorithm. For each $k \in [q+1]$, every job with source σ_i $(i \in S_k)$ will be set to be non-empty if the algorithm picks it (strictly) before time k, and empty otherwise. For each time t and $i \in [q^q]$, we denote by $\kappa_i(t)$ the number of machines such that the distance to σ_i is less than 1 at time t (i.e., machines at positions in $\{(i,x) \mid x \in (0,1]\}$). Define S_1 to be the set of indices $i \in [q^q]$ of end points with the $q^q - q^{q-1}$ largest values of $\kappa_i(1)$. Hence, $|S_1| = q^q - q^{q-1}$ and $\kappa_i(1) \geq \kappa_{i'}(1)$ for any $i \in S_1$ and $i' \notin S_1$. Similarly, for each $k = 2, 3, \dots, q$, define S_k sequentially to be the set of $i \in [q^q] \setminus \bigcup_{k'=1}^{k-1} S_{k'}$ such that $\kappa_i(k)$ $(i \in S_k)$ are the $q^{q+1-k} - p^{q-k}$ largest values among $\kappa_{i'}(k)$ $(i' \in [q^q] \setminus \bigcup_{k'=1}^{k} S_{k'})$. Finally, define S_{q+1} to be the set of remaining indices, i.e., $S_{q+1} = [q^q] \setminus \bigcup_{k'=1}^{q} S_{k'}$. We remark that S_k's are disjoint, and S_{q+1} is a singleton because $\sum_{k'=1}^{q} |S_{k'}| = \sum_{k'=1}^{q} (q^{q+1-k'} - q^{q-k'}) = q^q - 1$.

As some jobs with source σ_i $(i \in S_k)$ must be processed at time $q+1$ or later, the makespan of the schedule obtained by the online algorithm is at least $q+2$.

For each $k \in [q]$, let J_k be the set of non-empty jobs that were started to be processed in the interval $[k, k+1)$. We observe the cardinality of J_k. Recall that it takes one unit time to process a non-empty job. Hence, each machine can process at most one job in J_k. In addition, in order for a machine to start processing a job with source σ_i in the interval, the distance from σ_i from its state at time k must be less than 1. Since the sources of jobs in J_k are located at σ_i for some $i \in \bigcup_{k'=k+1}^{q+1} S_{k'}$, we obtain that

$$|J_k| \leq \sum_{i \in \bigcup_{k'=k+1}^{q+1} S_{k'}} \kappa_i(k) \leq m \cdot \frac{\left|\bigcup_{k'=k+1}^{q+1} S_{k'}\right|}{\left|\bigcup_{k'=k}^{q+1} S_{k'}\right|} = m \cdot \frac{q^{q-k}}{q^{q+1-k}} \cdot m = \frac{m}{q}.$$

As the algorithm starts to process any non-empty job at a time in $[1, q+1) = \bigcup_{k=1}^{q} [k, k+1)$, the total number of non-empty jobs is $\sum_{k \in [q]} |J_k| \leq (m/q) \cdot q = m$. Hence, the optimal offline makespan is at most the sum of 2 time units to process the empty jobs and 3 time units to process the non-empty jobs, which equals 5. Therefore, the competitive ratio is at least $(q+2)/5 = \Omega(\log m / \log\log m)$. ◀

5 Special Cases

In this section, we focus on the following three special cases: (i) single machine case $(m = 1)$, (ii) two machines case $(m = 2)$, and (iii) the optimal values of the 1-TSP and the m-TSP are close. In particular, for cases (ii) and (iii), we provide algorithms that improves Algorithm 1.

5.1 Single machine case

In this subsection, we consider the case where there is only one machine, i.e., $m = 1$. We first show that Algorithm 1 with $m = 1$ is 3-competitive.

▶ **Theorem 12.** *For any metric $\boldsymbol{M}$, Algorithm 1 is 3-competitive for the basic online $(\boldsymbol{M}, 1)$-scheduling problem.*

We remark that the competitive ratio 3 is tight for Algorithm 1. To see this, let us consider an instance with $\boldsymbol{M} = \mathbb{R}$ and three jobs, where job 1 is $(1,0,1)$, job 2 is $(1,1,0)$, and job 3 is $(0,1,1)$. In Algorithm 1, the machine processes the jobs in the order $1,2,3$, and the makespan is 6. On the other hand, the optimal offline makespan is 2 by processing jobs in the order $3,2,2$. Hence, the competitive ratio of Algorithm 1 is at least 3 when $m = 1$.

The proof of Theorem 12 implies that the makespan of the schedule obtained by Algorithm 1 is at most $2p(I) + \mathrm{LB}(I)$ for any instance I, i.e., $(\alpha,\beta,\gamma) = (0,2,1)$ for the values in (1). By Theorems 1, 3, and 5, the competitive ratios of IGNORE, REPLAN, and SMARTSTART incorporating with Algorithm 1 are 4.5, 5, and 4 for the online $(\boldsymbol{M},1)$-scheduling problem, respectively. We remark that our SMARTSTART is the same algorithm as the BOUNCER algorithm proposed by Lipmann et al. [19] (if it uses SMARTSTART as a 2-competitive algorithm for the online TSP over the sources).

▶ **Theorem 13.** *For any metric $\boldsymbol{M}$, there exists a 4-competitive algorithm for the online $(\boldsymbol{M},1)$-scheduling problem.*

Next, we provide a lower bound of the competitive ratio for the basic online $(\boldsymbol{M},1)$-scheduling problem.

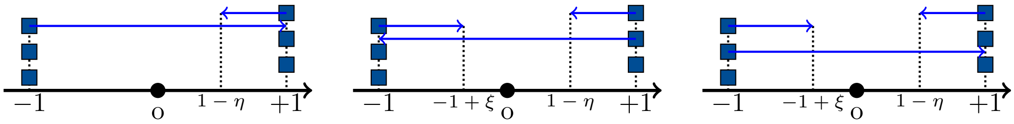
▶ **Theorem 14.** *There exists no 2.255-competitive algorithm even for the basic online $([-1,+1],\ 1)$-scheduling problem.*

Proof. Fixing an algorithm, we construct an adversary that gives the lower bound. Let η and ξ be the solutions of the following equations:

$$(8+2\eta)/4 = (10+2\xi)/(4+2\eta) = 12/(4+2\xi).$$

Note that $0.5101 < \eta < 0.5102$ and $0.6606 < \xi < 0.6607$.

Suppose that there are 6 jobs in total, three of which have their source state in $+1$, and the other three have their source state in -1. Without loss of generality, the algorithm first serves a job in $+1$. Then, the adversary sets the first job to be $(1, 1-\eta, \eta)$. We prove the theorem by cases according to the behavior of the algorithm (see Figure 2).



(a) Case 1. **(b)** Case 2.1. **(c)** Case 2.2.

Figure 2 Adversarial instance for the basic online $([-1,+1],1)$-scheduling problem.

Case 1. Suppose that the algorithm next serves another job in $+1$. Then the adversary sets the remaining jobs in $+1$ to be $(1,1,0), (1,1,0)$. After jobs in $+1$, the machine moves to -1, and the adversary sets the first job in -1 chosen by the algorithm to be $(-1,1,2)$, and the remaining jobs to be $(-1,-1,0), (-1,-1,0)$. Then the algorithm takes time at least $1 + \eta \times 2 + 2 + 2 + 2 + 1 = 8 + 2\eta$, while the optimal offline makespan is 4. Hence, the competitive ratio is at least $(8+2\eta)/4 > 2.255$.

Case 2. Suppose that the algorithm next serves a job in -1. Then, the adversary sets the job to be $(-1, -1+\xi)$. We divide cases according to the third job which the machine processes.

Case 2.1. If the third job is in -1, then the remaining jobs are set to be $(-1, -1, 0)$, $(-1, -1, 0)$, $(1, -1, 2)$, $(1, 1, 0)$, and the algorithm is made to choose the jobs in this order. The algorithm takes time at least $10 + 2\xi$ while the optimal offline makespan is $4 + 2\eta$. Hence, the competitive ratio is at least $(10 + 2\xi)/(4 + 2\eta) > 2.255$.

Case 2.2. If the third job is in $+1$, then the remaining jobs are set to be $(1, 1, 0)$, $(1, 1, 0)$, $(-1, 1, 2)$, $(-1, -1, 0)$, and the algorithm is made to choose the jobs in this order. The algorithm takes time at least 12 while the optimal offline makespan is $4 + 2\xi$. Hence, the competitive ratio is at least $12/(4 + 2\xi) > 2.255$.

Therefore, the competitive ratio is at least 2.255 in any cases. ◀

Combining the instance in the proof of Theorem 14 with the idea of setting jobs used in [19, Theorem 4], we can obtain a lower bound of 3.181 for the general case. This result also improves the best known lower bound of $1 + \frac{3\sqrt{2}}{2} \approx 3.12$ for the online dial-a-ride problem under the incomplete information model [19].

▶ **Theorem 15.** *For some metric $\boldsymbol{M}$, there exists no 3.181-competitive algorithm for the online $(\boldsymbol{M}, 1)$-scheduling problem.*

Proof. Let $\eta = 0.362\ldots$, $\xi = 0.514\ldots$, and $\rho = 3.181\ldots$ be the unique numbers satisfying

$$\rho = \frac{12 + 2 \cdot \eta}{4} = \frac{14 + 2 \cdot \xi}{4 + 2 \cdot \eta} = \frac{16}{4 + 2 \cdot \xi}. \tag{4}$$

Let $N \in \mathbb{N}$ be an integer bigger than $1.25/(\rho - 3.181)$, and let $\boldsymbol{M}$ be the star graph with $2 \cdot N$ leaves, each of which is at unit distance from the origin o. Consider an online algorithm ALG. We construct an instance for which ALG has makespan 3.181 times longer than the optimal (offline) schedule. Initially, there are three jobs (whose sources are) in each leaf. Each time ALG processes a job before time $4 \cdot N - 4$, the adversary sets the job to be empty, and adds a job with the same source and the release time being one unit amount of time later. Thus, at most $4 \cdot N - 4$ jobs are added by time $4 \cdot N - 4$, and there are exactly three unprocessed jobs (including unreleased jobs) in each leaf at time $4 \cdot N - 4$. We call them the *decisive* jobs (as in [19]), and specify their processing information below.

Because it takes two units of time to travel between leaves, at least two of the $2 \cdot N$ leaves, say 0^- and 0^+, are left unvisited by ALG during the time interval $[0, 4 \cdot N - 4)$. Likewise, among the $2 \cdot N - 2$ remaining leaves, there are two, say 1^- and 1^+, that are unvisited during $[4, 4 \cdot N - 4)$. Continuing in this way, we can name the leaves $0^\pm$, $1^\pm$, $\ldots$, $(N - 1)^\pm$ so that for each $j \in \{0, \ldots, N - 1\}$, neither j^- nor j^+ is visited by ALG during $[4 \cdot j, 4 \cdot N - 4)$. We identify the path $M_j \subseteq \boldsymbol{M}$ between j^- and j^+ (through o) with the interval $[-1, +1]$, and determine the processing information of the six decisive jobs in $j^\pm$ by the order in which ALG processes them, in the way described in the cases 1, 2.1, and 2.2 in the proof of Theorem 14 (but now with the new η and ξ defined in (4)).

Note that some jobs may be unreleased at time $4N - 3$. This affects Case 1 in the proof of Theorem 14, in which the machine processes three jobs in the same leaf consecutively. However, since the machine skips unreleased jobs and has to come back later, the completion time gets longer. Thus, we may assume that three jobs are released on each leaf.

By the analysis there, the amount of time after $4 \cdot N - 5$ spent by ALG in M_j is at least $8 + 2 \cdot \eta$, $10 + 2 \cdot \xi$, and 12 in the three cases, respectively, whereas the best schedule for the basic $(M_j, 1)$-scheduling instance given by the six decisive jobs in $j^\pm$ (forgetting their release times) has makespan 4, $4 + 2 \cdot \eta$, and $4 + 2 \cdot \xi$, respectively. Note that simply concatenating these best schedules for $j = 0, \ldots, N - 1$ gives an offline schedule for our whole instance, since all jobs in $j^{\pm}$ appear by time $4 \cdot j + 1$. Thus, writing N_1, $N_{2.1}$, and $N_{2.2}$ for the number of $j \in \{0, \dots, N-1\}$ for which the three cases happen ($N_1 + N_{2.1} + N_{2.2} = N$), we can bound from below the ratio of ALG's makespan to the optimal by

$$\begin{aligned}
&\frac{(4 \cdot N - 5) + N_1 \cdot (8 + 2 \cdot \eta) + N_{2.1} \cdot (10 + 2 \cdot \xi) + N_{2.2} \cdot 12}{N_1 \cdot 4 + N_{2.1} \cdot (4 + 2 \cdot \eta) + N_{2.2} \cdot (4 + 2 \cdot \xi)} \\
&= \frac{N_1 \cdot (12 + 2 \cdot \eta) + N_{2.1} \cdot (14 + 2 \cdot \xi) + N_{2.2} \cdot 16 - 5}{N_1 \cdot 4 + N_{2.1} \cdot (4 + 2 \cdot \eta) + N_{2.2} \cdot (4 + 2 \cdot \xi)} \\
&\geq \min\left\{ \frac{12 + 2 \cdot \eta}{4}, \frac{14 + 2 \cdot \xi}{4 + 2 \cdot \eta}, \frac{16}{4 + 2 \cdot \xi} \right\} - \frac{5}{N \cdot 4} \geq \rho - \frac{1.25}{N} > 3.181. \qquad \blacktriangleleft
\end{aligned}$$

5.2 Two machines case

In this subsection, we focus on the two-machine case, i.e., $m = 2$. As shown in Example 22 in Appendix, the competitive ratio of Algorithm 1 is at least 4 even for the basic online $(\mathbb{R}, 2)$-scheduling problem. In fact, we can improve Algorithm 1 as follows. The main idea is to move the machines in the opposite directions in phase 2. Formally, our algorithm can be stated as follows. The algorithm first computes optimal 2-TSP tours C_1 and C_2 over the sources. Then, machine i travels along C_i in a direction ($i = 1, 2$). When a machine arrives at the source of an unprocessed job, then it processes the job, returns to its source, and continues the travel. If C_1 is completed first, then machine 1 now travels along C_2 in the reverse direction. Similarly for the case when C_2 is completed. When all jobs have been processed, each machine directly returns to the origin as soon as possible.

▶ **Theorem 16.** *For any metric $\boldsymbol{M}$, there exists a 3.5-competitive algorithm for the basic online $(\boldsymbol{M}, 2)$-scheduling problem.*

We can also prove that the makespan of the above schedule is at most $\frac{1}{2}\mathrm{OPT}(I) + 2 \cdot p(I)/2 + \mathrm{LB}(I)$, i.e., $(\alpha, \beta, \gamma) = (1/2, 2, 1)$ for the values in (1). Thus, the competitive ratio of SMARTSTART incorporating with the above algorithm is $4 + \frac{\sqrt{3}}{2} \approx 4.866$ by Theorem 5.

▶ **Theorem 17.** *For any metric $\boldsymbol{M}$, there exists a $(4 + \frac{\sqrt{3}}{2})$-competitive algorithm for the online $(\boldsymbol{M}, 2)$-scheduling problem.*

5.3 Special metrics

Finally, we improve Algorithm 1 for the case where the optimal values of 1-TSP and m-TSP are close. A typical example of such a situation is instances with the half-line metric: the optimal value of 1-TSP coincides with that of m-TSP for any m because the values are twice the distance between origin and the rightmost source. The main idea of the improvement is to use the optimal 1-TSP tour instead of the m-TSP. This reduces the completion time because the number of phases is reduced to one. We further reduce the completion time by moving the machines in both directions of the optimal 1-TSP tour.

To be more precise, our algorithm first computes an optimal 1-TSP tour C over the sources. After that, half the machines (i.e., $\lceil m/2 \rceil$ machines) travel along the tour in a direction, and the other half (i.e., $\lfloor m/2 \rfloor$ machines) travel along the tour in the opposite direction. When a machine arrives at the source of an unprocessed job, then it processes the job, returns to its source, and continues the travel. When all jobs have been processed (including processing), each machine directly returns to the origin as soon as possible.

▶ **Theorem 18.** *Fix the number of machines $m \geq 2$ and the metric $\boldsymbol{M}$. Suppose that the ratio between the optimal values of* 1*-TSP and* m*-TSP is at most μ for any instance on $\boldsymbol{M}$. There exists a* $(\frac{\lceil m/2 \rceil}{m} \cdot \mu + 3)$*-competitive algorithm for the online* $(\boldsymbol{M}, m)$*-scheduling problem.*

Note that $\frac{\lceil m/2 \rceil}{m} \leq 2/3$ for any $m \geq 2$. Additionally, recall that, when $m = 1$, there is a 3-competitive algorithm for any metric (Theorem 13). As the value μ can be taken as 1 and 2 for $\mathbb{R}_+$ and $\mathbb{R}$, respectively, we can obtain the following corollaries.

▶ **Corollary 19.** *For any m, there exists a* $11/3$*-competitive algorithm for the basic online* $(\mathbb{R}_+, m)$*-scheduling problem.*

▶ **Corollary 20.** *For any m, there exists a* $13/3$*-competitive algorithm for the basic online* $(\mathbb{R}, m)$*-scheduling problem.*

6 Other Settings

In this section, we discuss other variants of online $(\boldsymbol{M}, m)$-scheduling problems.

We first consider minimizing the total completion time, i.e., the sum of completion times of all jobs. We observe that if the objective is to minimize the total completion time, any online algorithm is not competitive even for the basic online $(\mathbb{R}^0, 1)$-scheduling problem. Fix an online algorithm. Let n be a positive integer and consider an instance with n jobs with source and destination being the origin o. Suppose that the algorithm first processes job j^*. We set $(a_j, b_j, p_j) = (o, o, 0)$ for all $j \neq j^*$ and $(a_{j^*}, b_{j^*}, p_{j^*}) = (o, o, 1)$. Then, the total completion time of the algorithm is at least n as the completion time of every job is at least 1, but the optimal total completion time is 1 by processing job j^* last. As n can be taken as an arbitrarily large number, any algorithm is not competitive.

For the case where the processing time is known (but the destination state is not), we can design a constant competitive algorithm. In fact, we can obtain a 3-competitive algorithm for the basic case by using a solution of m-TSP in which the processing time is taken into account. On the other hand, if the destination state is known (but the processing time is not), the competitive ratio is $\Theta(\log m/\log\log m)$ since the lower bound shown in Theorem 7 also holds in this setting.

Next, we observe the case where the machines do not need to return to the origin, i.e., the objective is to minimize the time until all jobs have been completed. Such a setting is called *open* or *nomadic*, whereas the setting of our problem is said to be *closed* or *homing*. It is not difficult to see that the optimal makespan for the open setting is not less than half of the optimal makespan for the closed setting. Hence, if there exists a ρ-competitive algorithm for the closed setting, then there exists a 2ρ-competitive algorithm for the open setting. By combining this with 8, we obtain an $O(\log m/\log\log m)$-competitive algorithm for the open version of the online $(\boldsymbol{M}, m)$-scheduling problem. For the open version of the basic online $(\boldsymbol{M}, 1)$-scheduling problem, we can obtain a 3-competitive algorithm by the same way as Algorithm 1 (but use the path TSP). In addition, for the open version of the online $(\boldsymbol{M}, 1)$-scheduling problem, we can obtain a 6-competitive algorithm by applying REPLAN.

Finally, we discuss the preemptive version, i.e., the machines are allowed to preempt jobs in any point and resume the job later. We can observe that the competitive ratio of the basic online $(\boldsymbol{M}, m)$-scheduling problem is lower bounded by $\Omega(\log m/\log\log m)$ even when the source and the destination states are the same for all jobs. To see this, we consider an adversary similar to the one shown in Theorem 7. Consider the same metric, sources and the destinations of the jobs, and partition of the end points $S_1, S_2, \ldots, S_{q+1}$. However, we set the processing time of each job j with source in S_k to be $\min\{t + \epsilon, 1\}$ for each $k \in [q + 1]$, where t is the total processing length of that job j has been processed by time k and $\epsilon > 0$. Then, the makespan of the schedule obtained by the online algorithm is at least $q+2$ while the optimal offline makespan is at most 5. Therefore, by setting $\epsilon \to 0$, the competitive ratio is at least $(q+2)/5 = \Omega(\log m/\log\log m)$.

References

1 Norbert Ascheuer, Sven O. Krumke, and Jörg Rambau. Online Dial-a-Ride Problems: Minimizing the Completion Time. In *Proceedings of STACS*, volume 1770, pages 639–650, 2000.

2 G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the On-Line Travelling Salesman. *Algorithmica*, 29(4):560–581, 2001.

3 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Competitive algorithms for the on-line traveling salesman. In *Proceedings of the 4th Workshop on Algorithms and Data Structures*, pages 206–217, 1995.

4 Alexander Birx. *Competitive analysis of the online dial-a-ride problem*. PhD thesis, Technische Universität Darmstadt, 2020.

5 Alexander Birx and Yann Disser. Tight analysis of the smartstart algorithm for online dial-a-ride on the line. *SIAM Journal on Discrete Mathematics*, 34(2):1409–1443, 2020.

6 Alexander Birx, Yann Disser, and Kevin Schewior. Improved bounds for open online dial-a-ride on the line. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 145, 2019.

7 Antje Bjelde, Jan Hackfeld, Yann Disser, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Miriam Schlöter, Kevin Schewior, and Leen Stougie. Tight Bounds for Online TSP on the Line. *ACM Transactions on Algorithms*, 17(1):1–58, 2021.

8 Michiel Blom, Sven O Krumke, Willem E de Paepe, and Leen Stougie. The online tsp against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

9 Vincenzo Bonifaci, Maarten Lipmann, and Leen Stougie. *Online multi-server dial-a-ride problems*. SPOR-Report : reports in statistics, probability and operations research. Technische Universiteit Eindhoven, 2006.

10 Esteban Feuerstein and Leen Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.

11 Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation Algorithms for Some Routing Problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

12 Giorgio Gambosi and Gaia Nicosia. On-line scheduling with setup costs. *Information Processing Letters*, 73(1–2):61–68, 2000.

13 R. L. Graham. Bounds for Certain Multiprocessing Anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

14 Dan Gusfield. Bounds for naive multiple machine scheduling with release times and deadlines. *Journal of Algorithms*, 5(1):1–6, 1984.

15 Leslie A. Hall and David B. Shmoys. Approximation schemes for constrained scheduling problems. In *Proceedings of Annual Symposium on Foundations of Computer Science*, pages 134–139, 1989.

16 Hongtao Hu, K. K.H. Ng, and Yichen Qin. Robust Parallel Machine Scheduling Problem with Uncertainties and Sequence-Dependent Setup Time. *Scientific Programming*, 2016, 2016.

17 Patrick Jaillet and Michael R. Wagner. Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. *Operations Research*, 56(3):745–757, 2008.

18 Sven O. Krumke. *Online optimization: Competitive analysis and beyond*. PhD thesis, Technische Universität Berlin, 2001.

19 Maarten Lipmann, Xiwen Lu, Willem E. de Paepe, Rene A. Sitters, and Leen Stougie. On-Line Dial-a-Ride Problems Under a Restricted Information Model. *Algorithmica*, 40(4):319–329, 2004.

20 David M Miller, Hui-Chuan Chen, Jessica Matson, and Qiang Liu. A hybrid genetic algorithm for the single machine scheduling problem. *Journal of Heuristics*, 5(4):437–454, 1999.

21 David B. Shmoys, Joel Wein, and David P. Williamson. Scheduling parallel machines on-line. *SIAM Journal on Computing*, 24(6):1313–1331, 1995.

A Omitted proofs and examples

▶ **Theorem 5.** *SMARTSTART is* $\frac{6\alpha+4\beta+4\gamma+1+\sqrt{(2\alpha+1)^2+8\gamma}}{4}$*-competitive for the online* $(\boldsymbol{M}, m)$*-scheduling problem by setting* $\theta = \frac{2\alpha+1+\sqrt{(2\alpha+1)^2+8\gamma}}{4}$.

Proof. We note that θ satisfies the equality $(2\alpha+1)\theta+\gamma = 2\theta^2$. We consider a non-empty instance I. Let S be the set of jobs processed in the last subschedule of the SMARTSTART algorithm, and let t_S be the time when the execution of the subschedule started.

Suppose that the machines are idle just before time t_S. In this case, $t_S = \theta \cdot \mathrm{LB}(S_0)$ and we have

$$\begin{aligned}\mathrm{SMARTSTART}(I) &= t_S + \mathrm{ALG}_0(S_0)\\ &\leq \theta \cdot \mathrm{LB}(S_0) + \alpha\mathrm{OPT}(S_0) + \beta \cdot p(S_0)/m + \gamma\mathrm{LB}(S_0)\\ &\leq (\theta+\alpha+\beta+\gamma)\mathrm{OPT}(I)\\ &= \frac{6\alpha+4\beta+4\gamma+1+\sqrt{(2\alpha+1)^2+8\gamma}}{4} \cdot \mathrm{OPT}(I).\end{aligned}$$

Next, suppose that the last subschedule is started immediately after the second last one. Let R be the set of jobs processed in the second last subschedule and let t_R be the time when the second last subschedule is started. Define $\lambda = \frac{1}{2} + \frac{\gamma}{2\theta}$ $(= \theta - \alpha)$. Then, we have

$$\begin{aligned}&\mathrm{SMARTSTART}(I)\\ &= t_R + \mathrm{ALG}_0(R_0) + \mathrm{ALG}_0(S_0)\\ &\leq t_R + (\alpha\mathrm{OPT}(R_0) + \beta \cdot p(R_0)/m + \gamma\mathrm{LB}(R_0)) + (\alpha+\gamma)\mathrm{OPT}(S_0) + \beta p(S_0)/m\\ &\leq (1+\gamma/\theta)t_R + (\alpha+\beta)\mathrm{OPT}(I) + (\alpha+\gamma-\lambda)\mathrm{OPT}(S_0) + \lambda\mathrm{OPT}(S_0)\\ &\leq (1+\gamma/\theta)t_R + (2\alpha+\beta+\gamma-\lambda)\mathrm{OPT}(I) + \lambda\mathrm{OPT}(S_0)\\ &= 2\lambda \cdot t_R + (2\alpha+\beta+\gamma-\lambda)\mathrm{OPT}(I) + \lambda\mathrm{OPT}(S_0),\end{aligned}$$

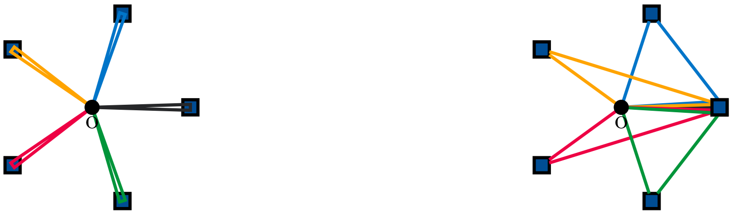
where the second inequality holds by $t_R \geq \theta \cdot \mathrm{LB}(R_0)$ and $p(R_0)/m + p(S_0)/m \leq \mathrm{OPT}(I)$, and the third inequality holds by $\alpha + \gamma \geq \lambda$. Let $f \in \arg\max_{j\in S} d(o, a_j)$. As $\mathrm{OPT}(S_0) \leq \mathrm{OPT}(I) - t_R + d(o, a_f)$, we have,

$$\begin{aligned}\mathrm{SMARTSTART}(I) &\leq 2\lambda \cdot t_R + (2\alpha+\beta+\gamma-\lambda)\mathrm{OPT}(I) + \lambda(\mathrm{OPT}(I) - t_R + d(o,a_f))\\ &= (2\alpha+\beta+\gamma)\mathrm{OPT}(I) + \lambda(t_R + d(o,a_f))\\ &\leq (2\alpha+\beta+\gamma+\lambda)\mathrm{OPT}(I) = (\alpha+\beta+\gamma+\theta)\mathrm{OPT}(I)\\ &= \frac{6\alpha+4\beta+4\gamma+1+\sqrt{(2\alpha+1)^2+8\gamma}}{4} \cdot \mathrm{OPT}(I).\end{aligned}$$ ◀

▶ **Example 21.** To observe this, consider an instance with m^2 jobs, where the source, destination, and processing time of job j are respectively

$$a_j = b_j = \left(\cos\tfrac{2\pi(j-1)}{m}, \sin\tfrac{2\pi(j-1)}{m}\right) \quad \text{and} \quad p_j = \begin{cases}100 & \text{if } j \equiv 1 \pmod{m},\\ 0 & \text{otherwise.}\end{cases}$$

Note that only jobs $1, m+1, \ldots, m(m-1)+1$ are non-empty. Then, the makespan of the schedule obtained by the simple method is $100m+2$. Indeed, each machine processes m jobs $i, m+i, \ldots, m(m-1)+i$ for some i. The completion time of the machine processing jobs $1, \ldots, m(m-1)+1$ is $100m+2$, while that of others is 2. On the other hand, the optimal schedule is that each machine processes each of the m non-empty jobs, and then machine i processes jobs $i, m+i, \ldots, m(m-1)+i$ for $i = 2, \ldots, m-1$ (see also Figure 3). The makespan of this schedule is at most 104, and hence the competitive ratio is $(100m+2)/104 = \Omega(m)$.



(a) Simple method. **(b)** Optimal.

Figure 3 An instance that the simple method does not work well ($m = 5$).

▶ **Theorem 12.** *For any metric $\boldsymbol{M}$, Algorithm 1 is 3-competitive for the basic online $(\boldsymbol{M}, 1)$-scheduling problem.*

Proof. When $m = 1$, Algorithm 1 just processes the jobs along with an optimal 1-TSP tour over the sources (there is only one phase). Let $h^{\text{sum}} = \sum_{j \in J}(p_j + d(b_j, a_j))$ and let ℓ_1 be the length of the optimal 1-TSP tour. By Lemma 10, we observe that $h^{\text{sum}} \leq 2p(I) \leq 2\text{OPT}(I)$ and $\ell_1 = \text{LB}(I) \leq \text{OPT}(I)$. Thus, the makespan of the schedule obtained by the algorithm is at most $\ell_1 + h^{\text{sum}} \leq \text{LB}(I) + 2p(I) \leq 3\text{OPT}(I)$. Therefore, Algorithm 1 is 3-competitive when $m = 1$. ◀

▶ **Example 22.** To observe this, let us consider an instance with 6 jobs, where job 1 is $(0, -1, 1)$, job 2 is $(1, 0, 1)$, job 3 is $(1, 1, 0)$, job 4 is $(0, 1, 1)$, job 5 is $(-1, 0, 1)$, and job 6 is $(-1, -1, 0)$. It is not difficult to see that an optimal 2-TSP tours over the sources are $C_1 = (1, 2, 3, 4)$ and $C_2 = (5, 6)$. In Algorithm 1, machines 1 and 2 processes jobs $\{1, 2, 3, 4\}$ and $\{5, 6\}$ in these orders (see Figure 4). Hence, the makespan of the schedule obtained by the algorithm is 8. On the other hand, the makespan is 2 if machines 1 and 2 processes jobs $\{4, 3, 2\}$ and $\{1, 6, 5\}$, respectively, in these orders. Hence, the competitive ratio of Algorithm 1 is at least 4 when $m = 2$.

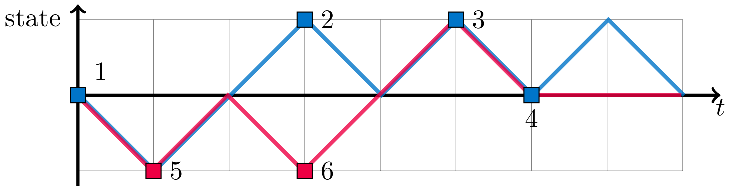


Figure 4 The schedule of Algorithm 1.

▶ **Theorem 16.** *For any metric $\boldsymbol{M}$, there exists a 3.5-competitive algorithm for the basic online $(\boldsymbol{M}, 2)$-scheduling problem.*

Proof. We analyze the above algorithm. Define ℓ_1 and ℓ_2 to be the lengths of the tours C_1 and C_2, respectively, and let $h^{\mathrm{sum}} = \sum_{j\in J}\bigl(p_j + d(b_j, a_j)\bigr)$. By Lemma 10, $(\ell_1 + \ell_2)/2 \leq \max\{\ell_1, \ell_2\} \leq \mathrm{OPT}(I)$ and $h^{\mathrm{sum}} \leq 2\sum_{j\in J} p_j \leq 4\mathrm{OPT}(I)$. Let τ^* be the latest time to start processing a job, and let i be the machine that returns to the origin later (breaking ties arbitrarily). Let job j be the last job processed by machine i. Note that job j may be started to be processed earlier than τ^*. We may assume without loss of generality that such a job exists, because otherwise i processes no jobs and the makespan is at most $\max\{\ell_1, \ell_2\} \leq \mathrm{OPT}(I)$. Note that $\tau^* \leq \frac{1}{2}\bigl(\ell_1 + \ell_2 + h^{\mathrm{sum}}\bigr) \leq 3\mathrm{OPT}(I)$.

Suppose that i is processing (or starts processing) job j at time τ^*. Then, the time when job j is started to process is at the latest $\frac{1}{2}\Bigl(\ell_1 + \ell_2 + \sum_{j'\in J\setminus\{j\}}\bigl(p_{j'} + d(b_{j'}, a_{j'})\bigr)\Bigr)$. Hence, the makespan is at most

$$\begin{aligned}
&\frac{1}{2}\left(\ell_1 + \ell_2 + \sum_{j'\in J\setminus\{j\}}\bigl(p_{j'} + d(b_{j'}, a_{j'})\bigr)\right) + p_j + d(b_j, o)\\
&\quad\leq \frac{1}{2}\bigl(\ell_1 + \ell_2 + h^{\mathrm{sum}} - p_j - d(b_j, a_j)\bigr) + p_j + \frac{d(o, a_j) + d(a_j, b_j) + d(b_j, o)}{2}\\
&\quad\leq \frac{1}{2}\bigl(\ell_1 + \ell_2 + h^{\mathrm{sum}} + d(o, a_j) + p_j + d(b_j, o)\bigr) \leq \frac{7}{2}\mathrm{OPT}(I),
\end{aligned}$$

where the last inequality holds by $d(o, a_j) + p_j + d(b_j, o) \leq \mathrm{OPT}(I)$.

Next, suppose that i is returning to a tour after processing job j at time τ^*. Let u be the state of i at time τ^*. As u is on the way from b_j to a_j, we have

$$\begin{aligned}
d(u, o) &\leq \min\{d(u, a_j) + d(a_j, o),\ d(u, b_j) + d(b_j, o)\}\\
&\leq \frac{1}{2}\Bigl(\bigl(d(u, a_j) + d(a_j, o)\bigr) + \bigl(d(u, b_j) + d(b_j, o)\bigr)\Bigr)\\
&= \frac{1}{2}\bigl(d(o, a_j) + d(a_j, b_j) + d(b_j, o)\bigr) \leq \frac{1}{2}\mathrm{OPT}(I).
\end{aligned}$$

Hence, the makespan is at most $\tau^* + d(u, o) \leq \frac{7}{2}\mathrm{OPT}(I)$.

Finally, suppose that i is traveling a tour without processing a job at time τ^*. Let u be the state of i at time τ^*. Then, $d(u, o) \leq \frac{1}{2}\max\{\ell_1, \ell_2\} \leq \frac{1}{2}\mathrm{OPT}(I)$. Hence, the makespan is at most $\tau^* + d(u, o) \leq \frac{7}{2}\mathrm{OPT}(I)$.

Therefore, the competitive ratio of the above algorithm is at most $7/2 = 3.5$. ◀

▶ **Theorem 18.** *Fix the number of machines $m \geq 2$ and the metric $\boldsymbol{M}$. Suppose that the ratio between the optimal values of 1-TSP and m-TSP is at most μ for any instance on $\boldsymbol{M}$. There exists a $(\frac{\lceil m/2\rceil}{m}\cdot\mu + 3)$-competitive algorithm for the online $(\boldsymbol{M}, m)$-scheduling problem.*

Proof. We analyze the above algorithm. Let ℓ_1^* and ℓ_m^* be the optimal length of the 1-TSP and m-TSP over the sources, respectively. In addition, let $h^{\mathrm{sum}} = \sum_{j\in J}(p_j + d(b_j, a_j))$, and let τ^* be the time when the last job started to be processed. Then, τ^* is at most

$$\frac{1}{m}(\lceil m/2\rceil\cdot\ell_1^* + h^{\mathrm{sum}}) \leq \frac{\lceil m/2\rceil}{m}\frac{\ell_1^*}{\ell_m^*}\mathrm{OPT}(I) + \mathrm{OPT}(I) \leq \left(\frac{\lceil m/2\rceil}{m}\cdot\mu + 2\right)\mathrm{OPT}(I)$$

because $\ell_m^* \leq \mathrm{OPT}(I)$ and $h^{\mathrm{sum}}/(2m) \leq \mathrm{OPT}(I)$ by Lemma 10. Next, consider the machine i^* that is the last to return to the origin. If i^* is either in the middle of processing job j or returning to its source after processing job j at time (just after) τ^*, the makespan is at most

$\tau^* + p_j + d(b_j, o) \le \left(\frac{\lceil m/2 \rceil}{m} \cdot \mu + 3\right) \mathrm{OPT}(I)$ because $p_j + d(b_j, o) \le d(o, a_j) + p_j + d(b_j, o) \le \mathrm{OPT}(I)$. Otherwise, suppose that the state of i^* at time τ^* is q, which is located on the way from a_j to $a_{j'}$. Then, we have

$$\begin{aligned} d(p, o) &\le \min\{d(q, a_j) + d(a_j, o),\ d(q, a_{j'}) + d(a_{j'}, o)\} \\ &\le \frac{1}{2}(d(a_j, o) + d(a_j, a_{j'}) + d(a_{j'}, o)) \le d(a_j, o) + d(a_{j'}, o) \le \mathrm{OPT}(I). \end{aligned}$$

Thus, the makespan is at most $\tau^* + \mathrm{OPT}(I) \le \left(\frac{\lceil m/2 \rceil}{m} \cdot \mu + 3\right) \mathrm{OPT}(I)$. ◀

A Simplicial Model for $\mathrm{KB4_n}$: Epistemic Logic with Agents That May Die

Éric Goubault
LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, France

Jérémy Ledent
MSP Group, University of Strathclyde, Glasgow, Scotland

Sergio Rajsbaum
National Autonomous University of Mexico, Mexico City, Mexico

Abstract

The standard semantics of multi-agent epistemic logic $\mathbf{S5_n}$ is based on Kripke models whose accessibility relations are reflexive, symmetric and transitive. This one dimensional structure contains implicit higher-dimensional information beyond pairwise interactions, that we formalized as pure simplicial models in a previous work in *Information and Computation* 2021 [10]. Here we extend the theory to encompass simplicial models that are not necessarily pure. The corresponding class of Kripke models are those where the accessibility relation is symmetric and transitive, but might not be reflexive. Such models correspond to the epistemic logic $\mathbf{KB4_n}$. Impure simplicial models arise in situations where two possible worlds may not have the same set of agents. We illustrate it with distributed computing examples of synchronous systems where processes may crash.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases Epistemic logic, Simplicial complexes, Distributed computing

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.33

Related Version *Full Version*: `https://arxiv.org/abs/2108.10293`

Funding Supported by UNAM-PAPIIT IN106520
Éric Goubault: This author was partially funded by AID project "Swarm of drones".
Sergio Rajsbaum: This work was partially done while this author was at École Polytechnique, France.

1 Introduction

A very successful research programme of using epistemic logic to reason about multi-agent systems began in the early 1980's showing the fundamental role of notions such as common knowledge [6, 20]. The semantics used is the one of "normal modal logics", based on the classic *possible worlds* relational structure developed by Rudolf Carnap, Stig Kanger, Jakko Hintikka and Saul Kripke in the late 1950's and early 1960's.

From global states to local states. The intimate relationship between distributed computing and algebraic topology discovered in 1993 [2, 13, 23] showed the importance of moving from using worlds as the primary object, to *perspectives* about possible worlds. After all, what exists in many distributed systems is only the local states of the agents and events observable within the system.

Taking local states as the main notion led to the study of distributed systems based on geometric structures called *simplicial complexes*. In this context, a simplicial complex is constructed using the local states as vertices and the global states as simplexes. While the solvability of some distributed tasks such as *consensus* depends only on the one-dimensional (graph) connectivity of global states, the solvability of other tasks, most notably *k-set* *agreement*, depends on the higher-dimensional connectivity of the simplicial complex of local states. See [12] for an overview of the topological theory of distributed computability.

Pure simplicial model semantics [10]. From the very beginning [23], distributed computer scientists have used the word "knowledge" informally to explain their use of simplicial complexes. However, a formal link with epistemic logic was established only recently [10].

The idea is to replace the usual one-dimensional Kripke models by a new class of models based on simplicial complexes, called *simplicial models*. In [10], we focused on modelling the standard multi-agent epistemic logic, $\mathbf{S5_n}$. In this setting, a core assumption is that the same set of n agents always participate in every possible world. Because of this, all the facets of the simplicial model are of the same dimension. Such models are called *pure* simplicial models. With this restriction, we showed that the class of pure simplicial models is equivalent to the usual class of $\mathbf{S5_n}$ Kripke models.

Using pure simplicial models, we provided epistemic logic tools to reason about solvability of distributed tasks such as consensus and approximate agreement. In subsequent work, we also studied the equality negation task, explored bisimilarity of pure simplicial models, and connections with covering spaces [4, 9, 25]. In [10], we left open the question of a logical obstruction to the solvability of k-set agreement, which was later given by Yagi and Nishimura [27] using the notion of distributed knowledge [11], in a sense a higher-dimensional version of knowledge.

Systems with detectable crashes. In this paper, we wish to extend the work of [10] by lifting the restriction to "pure" simplicial complexes. In distributed computing, pure complexes can be used to analyse the basic *wait-free* shared-memory model of computation [14]. However, impure[1] complexes also show up in many situations: perhaps the most simple one is the *synchronous crash model*, where processes may fail by crashing[2]. Due to the synchronous nature of the system, when a process crashes, the other processes will eventually know about it. This contrasts with asynchronous systems, where processes can be arbitrarily slow, and there is no way to distinguish a crashed process from a slow one.

Systems where crash-prone processes operate in synchronous rounds have been thoroughly studied since early on in distributed computing, see e.g. [7, 17]. At the start of each round, every process sends a message to all the other processes, in unspecified order. A process may crash at any time during the round, in which case only a subset of its messages will be received. A global clock indicates the end of the round: any message that has not been received by then signifies that the sender has crashed. Moreover, we usually assume a *full-information* protocol: in each round, the messages sent by the processes consist of its local state at the end of the previous round.

Figure 1 below depicts the simplicial complexes of local states for three processes, after one and two rounds of the synchronous crash model. In the initial situation (left), the local states are binary input values of the processes, 0 or 1. Each of the 8 triangles represents a possible global state, i.e. an assignment of inputs to processes. The two other complexes (middle and right) represent the situation after one round and two rounds, respectively. These complexes are impure: they contain both triangles (representing global states where all three

[1] Throughout this paper, the adjective "impure" usually stands for "not necessarily pure".

[2] In the distributed computing literature, agents are called *processes*, and when a process stops its execution prematurely, it is said to have *crashed*. In this paper, we will say that agents may *die*.

processes are alive) and edges (representing global states where only two agents are alive). Throughout the paper, we use this model as a running example, starting with Example 6. Further details from the distributed computing perspective can be found in [15].

Synchronous systems have also been studied using epistemic logic, e.g. in the seminal work of Dwork and Moses [5], where a complete characterization of the number of rounds required to reach simultaneous consensus is given, in terms of common knowledge. The focus however has been on studying solvability of consensus and other problems related to common knowledge, which as mentioned above, depend only on the 1-dimensional connectivity of epistemic models.

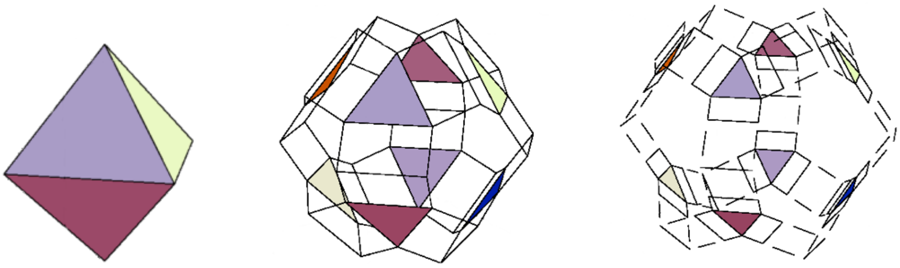


Figure 1 Input complex for three agents starting with binary inputs, then the complex after one, and after two rounds. At most one agent may die [15].

Contributions. With the long-term goal of going beyond consensus-like problems, to k-set agreement, renaming, and other tasks whose solvability depends on higher dimensional topological connectivity, we introduce in this paper an epistemic logic where agents may die, whose semantics is naturally given by impure simplicial models.

Our approach is guided by the categorical equivalence between $\mathbf{S5_n}$ Kripke models and pure simplicial models, established in [10]. It is easy and natural to generalize the class of simplicial models by simply removing the "pure" assumption. However, the main technical challenge resides in finding an equivalent category of Kripke models. This is achieved in Section 3, where the categorical equivalence is established in Theorem 23 for the frames, and Theorem 27 for the models. Guided by the equivalence with simplicial models, we introduce *partial epistemic models*, whose underlying frame has the following characteristics:

- Indistinguishability relations must be transitive and symmetric, but may not be reflexive.
- The frames must be *proper*, in a sense defined in Section 3.1.

Surprisingly, the morphisms between those frames are also unusual: a world is mapped to a sets of worlds, which must be *saturated* (Definition 13).

In Section 4, we reap the benefits of this equivalence theorem. Modal logics on Kripke models are well understood, and we can then translate results back to simplicial models. Each of the peculiar conditions that we impose on partial epistemic frames reveals an implicit assumption of simplicial models.

The consequence of losing reflexivity is that the logic is no longer $\mathbf{S5_n}$, but instead $\mathbf{KB4_n}$, where the Axiom $\mathbf{T}$ does not hold. This logic is not often considered by logicians; its close cousin $\mathbf{KD45_n}$ being more commonly studied, in order to reason about belief [26]. But, as we argue in Sections 4.3 and 4.4, $\mathbf{KB4_n}$ is an interesting setting to reason about alive and dead agents. Moreover, the requirement of having proper frames leads us to introduce two additional axioms: the axiom of Non-Emptiness $\mathbf{NE}$ says that at least one agent is alive in every world; and the Single-Agent axioms $\mathbf{SA_a}$ says that if exactly one agent a is alive, this agent knows that all other agents are dead. In Section 4.5, we claim that the logic $\mathbf{KB4_n}$ augmented with these two extra axioms is sound and complete with respect to class of (possibly non-pure) simplicial models. While soundness is easy to prove, the proof of completeness is more intricate and we leave it for the full version of this work. Finally in Section 4.6, we prove the so-called *knowledge gain* property, which has been instrumental in applications to impossibility results in distributed computing, see e.g. [10].

Related work. A line of work started by Dwork and Moses [5] studied in great detail the synchronous crash failures model from an epistemic logic perspective. However, in their approach, the crashed processes are treated the same as the active ones, with a distinguished local state "`fail`". In that sense, all agents are present in every state, hence they still model the usual epistemic logic $\mathbf{S5_n}$. Instead of changing the underlying Kripke models as we do here, they introduce new knowledge and common knowledge operators that take into account the non-rigid set of agents (see e.g. [22], Chapter 6.4).

Giving a formal epistemic semantics to impure simplicial models has also been attempted by van Ditmarsch [24], at the same time and independently from our work. This approach end up quite different from ours. It describes a two-staged semantics with a *definability relation* prescribing which formulas can be interpreted, on top of which the usual *satisfaction relation* is defined. This results in a quite peculiar logic: for instance, it does not obey Axiom **K**, which is the common ground of all Kripke-style modal logics. The question of finding a complete axiomatization is left open. In contrast, we take a more systematic approach: we first establish a tight categorical correspondence between simplicial models and Kripke models. Via this correspondence, we translate the standard Kripke-style semantics to simplicial models. This leads us to the modal logic $\mathbf{KB4_n}$. We will discuss further the technical differences between our approach and that of [24] in Section 3.2.

2 Background on simplicial complexes and Kripke structures

Chromatic simplicial complexes. Simplicial complexes are the basic structure of combinatorial topology [16]. In the field of fault-tolerant distributed computing [12], their vertices are usually labelled by process names, often viewed as colours; hence the adjective "chromatic".

▶ **Definition 1.** *A* simplicial complex *is a pair $\mathcal{C} = \langle V, S\rangle$ where V is a set, and $S \subseteq \mathscr{P}(V)$ is a family of non-empty subsets of V such that for all $v \in V$, $\{v\} \in S$, and S is downward-closed: for all $X \in S$, if Y is non-empty and $Y \subseteq X$ then $Y \in S$.*

Considering a finite, non-empty set A of agents, *a* chromatic simplicial complex *coloured by A is a triple $\langle V, S, \chi\rangle$ where $\langle V, S\rangle$ is a simplicial complex, and $\chi : V \to A$ assigns colours to vertices such that for every $X \in S$, all vertices of X have distinct colours.*

Elements of V are called *vertices*, and are identified with singletons of S. Elements of S are *simplexes*, and the ones that are maximal w.r.t. inclusion are *facets*. The set of facets of $\mathcal{C}$ is written $\mathcal{F}(\mathcal{C})$. The *dimension* of a simplex $X \in S$ is $\dim(X) = |X| - 1$. A simplicial complex C is *pure* if all facets are of the same dimension. The condition of having distinct colours for vertices of the same simplex is a fairly strong one: in particular, we will always be allowed to take the (unique) subface of a simplex X of a chromatic simplicial complex with colours in some subset U of $\chi(X)$.

▶ **Definition 2.** *A* chromatic simplicial map *$f : \mathcal{C} \to \mathcal{D}$ from $\mathcal{C} = \langle V, S, \chi\rangle$ to $\mathcal{D} = \langle V', S', \chi'\rangle$ is a function $f : V \to V'$ preserving simplexes, i.e. for every $X \in S$, $f(X) \in S'$, and preserving colours, i.e. for every $v \in V$, $\chi'(f(v)) = \chi(v)$.*

We denote by $\mathsf{SimCpx_A}$ the category of chromatic simplicial complexes coloured by A, and $\mathsf{SimCpx_A^{pure}}$ the full sub-category of pure chromatic simplicial complexes on A.

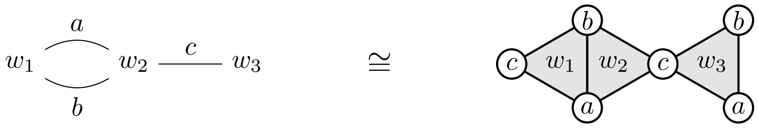
Equivalence with epistemic frames. The traditional possible worlds semantics of (multi-agent) modal logics relies on the notion of Kripke frame. Let A be a finite set of agents.

▶ **Definition 3.** *A* Kripke frame $M = \langle W, R\rangle$ *is a set of* worlds W*, together with an A-indexed family of relations on W, $R : A \to \mathscr{P}(W \times W)$. We write R_a rather than $R(a)$, and $u\,R_a\,v$ instead of $(u, v) \in R_a$. The relation R_a is called the* a-accessibility relation*. Given two Kripke frames $M = \langle W, R\rangle$ and $N = \langle W', R'\rangle$, a* morphism *from M to N is a function $f : W \to W'$ such that for all $u, v \in W$, for all $a \in A$, $u\,R_a\,v$ implies $f(u)\,R'_a\,f(v)$.*

To model multi-agent epistemic logic $\mathbf{S5_n}$, we additionally require each relation R_a to be an equivalence relation. When this is the case, we usually denote the relation by $\sim_a$, and call it the *indistinguishability relation.* For the equivalence class of w with respect to $\sim_a$, we write $[w]_a \subseteq W$. Kripke frames satisfying this condition are called *epistemic frames.* An epistemic frame is *proper* when two distinct worlds can always be distinguished by at least one agent: for all $w, w' \in W$, if $w \neq w'$ then $w \not\sim_a w'$ for some $a \in A$. In [10], we exploited an equivalence of categories between pure chromatic simplicial complexes and proper Kripke frames, to give an interpretation of $\mathbf{S5_n}$ on simplicial models. This allowed us to apply epistemic logics to study distributed tasks.

▶ **Theorem 4** ([10]). *The category of pure chromatic simplicial complexes* $\mathsf{SimCpx_A^{pure}}$ *is equivalent to the category of proper epistemic frames* $\mathsf{EFrame_A^{proper}}$.

▶ **Example 5.** The picture below shows an epistemic frame (left) and its associated chromatic simplicial complex (right). The three agents are named a, b, c. The three worlds $\{w_1, w_2, w_3\}$ of the epistemic frame correspond to the three facets (triangles) of the simplicial complex. In the epistemic frame, the c-labelled edge between the worlds w_2 and w_3 indicates that $w_2 \sim_c w_3$. Correspondingly, the two facets w_2 and w_3 of the simplicial complex share a common vertex, labelled by agent c. Similarly, the worlds w_1 and w_2 are indistinguishable by both agents a and b; so the corresponding facets share their ab-labelled edge.



3 Partial epistemic frames and simplicial complexes

In this section, we generalise Theorem 4 to deal with chromatic simplicial complexes that may not be pure. For that purpose, we will need to enlarge the class of Kripke frames to be considered, which we call *partial epistemic frames.* First, we start with our running example of an impure simplicial complex, which has been studied in distributed computing.

▶ **Example 6** (Synchronous crash-failure model, one round, three agents). Consider a set of three processes/agents $A = \{a, b, c\}$. For simplicity, we consider a single initial state where the agent a, b, c start with input value $1, 2, 3$, respectively[3]. Each agent sends a message to

[3] Typically, in distributed computing, many initial assignments of inputs are possible. Thus, we model a situation where the inputs of other processes are not known until a message from them is received.

the two other agents (and to itself, for uniformity), containing its input value. An agent may *crash* during the computation, in which case it stops sending messages. We assume moreover that at most two agents may crash, as in e.g. [5]. At the end of the round, an agent is *alive* if it successfully sent all its messages, and *dead* if it crashed before finishing. The *view* (or local state) of an alive agent is the set of messages that it received during the round. Note that an alive agent always sees its own value. For instance, the four possible views of agent a after one round are $\{1\}, \{1,2\}, \{1,3\}$ and $\{1,2,3\}$.

This situation is modelled by the chromatic simplicial complex $\mathcal{C}$ on the left of Figure 2. Formally, the vertices of $\mathcal{C}$ are pairs (a, view) where $a \in A$ and $\mathsf{view} \subseteq \{1,2,3\}$ is its view. There are 12 such vertices, 4 for each agent. The colouring $\chi(a, \mathsf{view}) = a$ of a vertex is indicated on the picture. There are 13 facets $w_0, \dots, w_{12}$, corresponding to the possible global states at the end of the round. The middle triangle $w_1 = \{(a, \mathsf{view}_a), (b, \mathsf{view}_b), (c, \mathsf{view}_c)\}$, with $\mathsf{view}_a = \mathsf{view}_b = \mathsf{view}_c = \{1,2,3\}$, represents the execution where no agent dies. The three isolated vertices, w_0, w_{11}, w_{12} are executions where two agents died. For instance, in $w_0 = \{(a, \{1\})\}$, both b and c crashed before sending their value to a. The 9 edges represent situations where one agent died, and two survived. For example, $w_2 = \{(a, \mathsf{view}_a), (c, \mathsf{view}_c)\}$, with $\mathsf{view}_a = \{1,2,3\}$ and $\mathsf{view}_c = \{1,3\}$, represents the execution where b crashed after sending its value to a, but not to c. In w_{10}, agent b crashed before sending any messages.

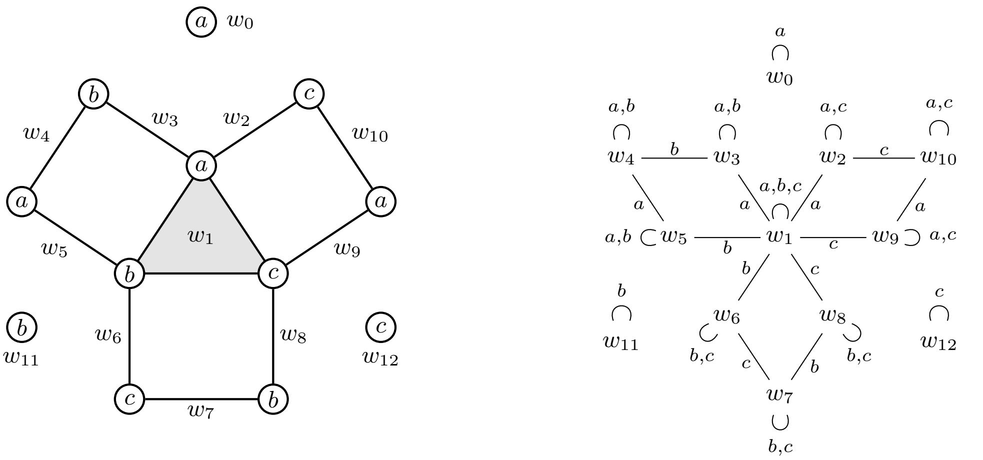


Figure 2 A chromatic simplicial complex $\mathcal{C}$ (left), and a proper partial epistemic frame M (right). The three agents are $A = \{a, b, c\}$ and the 13 facets/worlds are labelled $w_0, \dots, w_{12}$.

3.1 Partial epistemic frames

We consider now another type of Krikpe frame, in the spirit of PER semantic models of programming languages and “Kripke logical partial equivalence relations” of e.g. [18].

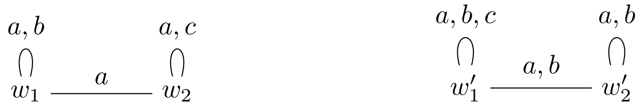
▶ **Definition 7.** *A* Partial Equivalence Relation *(PER) on a set X is a relation $R \subseteq X \times X$ which is symmetric and transitive (but not necessarily reflexive).*

The *domain* of a PER R is the set $\mathsf{dom}(R) = \{x \in X \mid R(x,x)\} \subseteq X$, and it is easy to see that R is an equivalence relation on its domain, and empty outside of it. Thus, PERs are equivalent to the “local equivalence relations” defined in [24]. Recall A is the set of agents.

▶ **Definition 8.** *A* partial epistemic frame *$M = \langle W, \sim \rangle$ is a Kripke frame such that each relation $(\sim_a)_{a \in A}$ is a PER.*

We say that agent a is *alive* in a world w when $w \in \mathsf{dom}(\sim_a)$, i.e., when $w \sim_a w$. In that case, we write $[w]_a$ for the equivalence class of w with respect to $\sim_a$, within $\mathsf{dom}(\sim_a)$. We write $\overline{w}$ for the set of agents that are alive in world w and $\underline{w}$ for the set of agents that are dead in world w (the complement of $\overline{w}$). A partial epistemic frame is *proper* if in all worlds, there is at least one agent which is alive, and moreover any two distinct worlds w, w' can be distinguished by at least one agent that is alive in w, i.e., $\forall w, w' \in W,\ \exists a \in A,\ \ w \sim_a w$ and $(w \neq w' \implies w \not\sim_a w')$. Note that, by symmetry of $\neq$, there is also a (possibly different) agent a' that is alive in w' and can distinguish w and w'.

▶ **Example 9.** Two partial epistemic frames over the set of agents $A = \{a, b, c\}$ are represented below. The frame on the left is proper, because agent b is alive in w_1 and can distinguish between w_1 and w_2; and agent c is alive in w_2 and can distinguish between w_1 and w_2. The frame on the right is not proper, because there is no agent alive in w_2' that can distinguish between w_1' and w_2'.



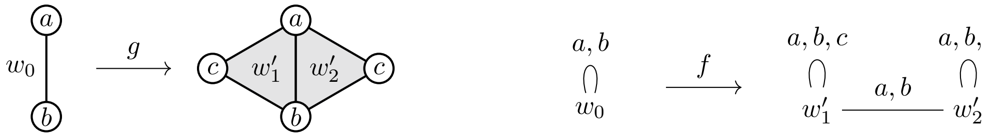
▶ **Example 10.** The partial epistemic frame modelling the synchronous crash model of Example 6 is pictured Figure 2 (right). It has 13 worlds $w_0, \ldots, w_{12}$. In each world, the set of alive agents can be read off the reflexive "loop" edge.

- In w_1, all three agents $\{a, b, c\}$ are alive.
- In worlds w_3, w_4 and w_5, the two alive agents are a and b. In worlds w_2, w_{10} and w_9, the alive agents are a and c. And in worlds w_6, w_7, w_8, agents b and c are alive.
- In w_0, only a is alive. In w_{11}, only b is alive, and in w_{12}, only c is alive.

The accessibility relation is represented by edges labelled with the agents that do not distinguish between the worlds at its extremities. For instance, agent a cannot distinguish between w_3 and w_1, and agent b cannot distinguish between w_3 and w_4. It can easily be checked to be a proper partial epistemic frame.

Morphisms of partial epistemic frames. Our notion of morphism for partial epistemic frames differs from the one for a general Kripke frame (Definition 3). Here again, our definitions are guided by our goal (Theorem 23), the equivalence between simplicial maps and morphisms of partial epistemic frames. Example 11 below should help motivate our definitions. The novelty arises when we want a morphism f that maps a world w, in which some agents $\overline{w}$ are alive, to a world w_1' where strictly more agents are alive. In this case, there might exist some other world w_2', such that $w_1' \sim_a w_2'$ for all $a \in \overline{w}$. We claim that such a world w_2' should also be in the image of w by the morphism f. Thus, $f(w)$ is not a world but a set of worlds, which we require to be *saturated*, in the following sense.

▶ **Example 11.** The two pictures below show a chromatic simplicial map g (left) and a morphism f of partial epistemic frames (right). The simplicial map g is uniquely specified by the preservation of colours: it maps the edge w_0 onto the vertical ab-coloured edge of the complex on the right. The morphism f is defined by $f(w_0) = \{w_1', w_2'\}$. We will see in Section 3.2 how to relate these morphisms: one can be built from the other, and vice-versa.



▶ **Definition 12.** *Given a partial epistemic frame $M = \langle W, \sim \rangle$, a subset of agents $U \subseteq A$, and a world $w \in W$, let $\mathsf{sat}_U(w) = \{w' \in W \mid w \sim_a w' \text{ for all } a \in U\}$.*

The saturation requirement will be crucial in Section 3.2 when we establish the equivalence of categories between partial epistemic frames and chromatic simplicial complexes.

▶ **Definition 13.** *Let $M = \langle W, \sim \rangle$ and $N = \langle W', \sim' \rangle$ be two partial epistemic frames. A morphism of partial epistemic frame from M to N is a function $f : W \to \mathscr{P}(W')$ such that*

- (Preservation of $\sim$) *for all $a \in A$, for all $u, v \in W$, $u \sim_a v$ implies $u' \sim'_a v'$, for all $u' \in f(u)$ and $v' \in f(v)$,*
- (Saturation) *for all $u \in W$, there exists $u' \in f(u)$ such that $f(u) = \mathsf{sat}_{\overline{u}}(u')$.*

Composition of morphisms is defined by $(g \circ f)(u) = \mathsf{sat}_{\overline{u}}(w)$, for some $v \in f(u)$ and $w \in g(v)$.

Let us check that the composite $g \circ f$ above is well-defined, i.e., that it does not depend on the choice of $v \in f(u)$ and $w \in g(v)$. Assume we pick $v' \in f(u)$ and $w' \in g(v')$ instead. Then $v \sim_a v'$ for all $a \in \overline{u}$, because $f(u)$ is saturated. And by preservation of $\sim$, we get $w \sim_a w'$ for all $a \in \overline{u}$, that is, $\mathsf{sat}_{\overline{u}}(w) = \mathsf{sat}_{\overline{u}}(w')$.

The first condition of a morphism f of partial epistemic frame above means that worlds that are indistinguishable by some agent a should have images composed of worlds that are indistinguishable by a. The second condition states that the image of a world u of M is "generated" by a world u' of N, as the set of all worlds of N that cannot be distinguished from u' by the agents alive in u. In particular, notice that the saturation condition implies that $f(u)$ is always non-empty.

The next proposition says that, on proper frames, the only case when $f(u)$ can be multivalued is when $\overline{u} \subsetneq \overline{u'}$ for every u' in $f(u)$.

▶ **Proposition 14.** *Let $M = \langle W, \sim \rangle$ and $N = \langle W', \sim' \rangle$ be two partial epistemic frames, and $f : M \to N$ be a morphism. For all $u \in W$ and $u' \in f(u)$, $\overline{u} \subseteq \overline{u'}$. Moreover, if N is proper and $\overline{u} = \overline{u'}$, then $f(u) = \{u'\}$.*

Proof. The first fact is a direct consequence of the preservation of $\sim$. For the second one, let $u' \in f(u)$ such that $\overline{u'} = \overline{u}$. Assume by contradiction that there is $u'' \in f(u)$ with $u'' \neq u'$. By saturation, we have $u'' \sim_a u'$ for all $a \in \overline{u} = \overline{u'}$. This is impossible since N is proper. ◀

The category of partial epistemic frames with set of agents A is denoted by KPER_A, and the full subcategory of proper partial epistemic frames is denoted by $\mathsf{KPER}_\mathsf{A}^{\text{proper}}$. Note that the category of proper epistemic frames $\mathsf{EFrame}_\mathsf{A}^{\text{proper}}$ is a full subcategory of $\mathsf{KPER}_\mathsf{A}^{\text{proper}}$. Indeed, in an epistemic frame all agents are alive in all worlds, so by Proposition 14 morphisms between proper epistemic frames are single-valued. Then Definition 13 reduces to the standard notion of Kripke frame morphisms (Definition 3).

3.2 Equivalence between chromatic simplicial complexes and partial epistemic frames

In this section, we show how to canonically associate a proper partial epistemic frame with any chromatic simplicial complex, and vice-versa. In fact, we have an equivalence of categories, meaning this correspondence can be extended to morphisms too (see Example 11). We construct functors $\kappa : \mathsf{SimCpx}_\mathsf{A} \to \mathsf{KPER}_\mathsf{A}^{\text{proper}}$ and $\sigma : \mathsf{KPER}_\mathsf{A}^{\text{proper}} \to \mathsf{SimCpx}_\mathsf{A}$ and show that they form an equivalence of categories in Theorem 23. A similar correspondence appears in [24], with two differences:

- They only show the equivalence between the objets of those categories, while we also deal with morphisms. To achieve this, we had to define morphisms of partial epistemic frames (Definition 13), since the standard notion does not work.
- They only show that $\kappa \circ \sigma(M)$ is bisimilar to M, while we prove a stronger result, that there is an isomorphism. To achieve this, we had to impose the condition of M being proper, which is not considered in [24].

▶ **Definition 15** (Functor κ). *Let $\mathcal{C} = \langle V, S, \chi \rangle$ be a chromatic simplicial complex on the set of agents A. Its associated partial epistemic frame is $\kappa(\mathcal{C}) = \langle W, \sim \rangle$, where $W := \mathcal{F}(\mathcal{C})$ is the set of facets of $\mathcal{C}$, and the PER $\sim_a$ is given by $X \sim_a Y$ if $a \in \chi(X \cap Y)$ (for $X, Y \in \mathcal{F}(\mathcal{C})$).*

The image of a morphism $f : \mathcal{C} \to \mathcal{D}$ in $\mathsf{SimCpx_A}$, is the morphism $\kappa(f) : \kappa(\mathcal{C}) \to \kappa(\mathcal{D})$ in $\mathsf{KPER_A^{proper}}$ that takes a facet $X \in \mathcal{F}(\mathcal{C})$ to $\kappa(f)(X) = \{Z \in \mathcal{F}(\mathcal{D}) \mid f(X) \subseteq Z\}$.

▶ **Example 16.** In Figure 2, the simplicial complex $\mathcal{C}$ on the left is mapped by κ to the partial epistemic frame $M = \kappa(\mathcal{C})$ on the right. The epistemic frame M contains a world per facet $w_0, \dots, w_{12}$ of the simplicial complex. The reflexive "loops" in the M, indicating which agents are alive in a given world, are labelled with the colours of the corresponding facet. For instance, $w_1 \sim_{\{a,b,c\}} w_1$ but $w_3 \sim_{\{a,b\}} w_3$ only; because w_3 in $\mathcal{C}$ is an edge whose extremities have colours a and b.

We now check that $\kappa(\mathcal{C})$ and $\kappa(f)$ above are correctly defined.

▶ **Proposition 17.** *κ is a well-defined functor from $\mathsf{SimCpx_A}$ to $\mathsf{KPER_A^{proper}}$.*

Conversely, we now consider a partial epistemic frame $M = \langle W, \sim \rangle$ on the set of agents A, and we define the associated chromatic simplicial complex $\sigma(M)$. Intuitively, each world $w \in W$ where $k+1$ agents are alive will be represented by a facet X_w of dimension k, whose vertices are coloured by $\overline{w}$. Such facets must then be "glued" together according to the indistinguishability relations. Formally, this is done by the following quotient construction:

▶ **Definition 18** (Functor σ on objects). *Let $M = \langle W, \sim \rangle$ be a partial epistemic frame. Its associated chromatic simplicial complex is $\sigma(M) = \langle V, S, \chi \rangle$, where:*

- *The set of vertices is $V = \{(a, [w]_a) \mid w \in W, a \in \overline{w}\}$. We denote such a vertex $(a, [w]_a)$ by v_a^w for succinctness; but note that $v_a^w = v_a^{w'}$ when $w \sim_a w'$.*
- *The facets are of the form $X_w = \{v_a^w \mid a \in \overline{w}\}$ for each $w \in W$; and the set S consists of all their sub-simplexes.*
- *The colouring is given by $\chi(v_a^w) = a$.*

It is straightforward to see that this is a chromatic simplicial complex. We now check that there is indeed one distinct facet of $\sigma(M)$ for each world of M.

▶ **Lemma 19.** *If M is proper, the facets of $\sigma(M)$ are in bijection with the worlds of M.*

▶ **Example 20.** In Figure 2, the partial epistemic frame M on the right is mapped by σ onto the simplicial complex $\mathcal{C} = \sigma(M)$ on the left. Each world $w_0, \dots, w_{12}$ of M is turned into a facet of the simplicial complex $\sigma(M)$, whose dimension is the number of alive agents minus one. These facets are glued along the sub-simplexes whose colours are the agents that cannot distinguish between two worlds. For instance, world w_1 is associated with the facet of the same name, with 3 colours, hence of dimension 2 (the central triangle). On the other hand, the world w_3 turns into an edge (dimension 1), glued to the triangle w_1 along the vertex with colour a, because $w_1 \sim_a w_3$.

We also define the action of σ on morphisms of partial epistemic frames:

▶ **Definition 21** (Functor σ on morphisms). *Now let $f : M \to N$ be a morphism in $\mathsf{KPER}_{\mathsf{A}}^{\mathrm{proper}}$. We define the simplicial map $\sigma(f) : \sigma(M) \to \sigma(N)$ as follows. For each vertex of $\sigma(M)$ of the form v_a^w with $w \in W$, we pick any $w' \in f(w)$ and define $\sigma(f)(v_a^w) = v_a^{w'}$.*

To check that this is well-defined, we need to show that the simplicial map $\sigma(f)$ does not depend on the choices of w and w'. Assume we pick a different world $u' \in f(w)$, $u' \neq w'$. By the saturation property of f we have $u' \sim'_a w'$, so $v_a^{u'} = v_a^{w'}$. Hence $\sigma(f)(v_a^w)$ is a uniquely defined vertex of $\sigma(N)$. Now, assume that the vertex v_a^w of $\sigma(M)$ could also be described as v_a^u with $u \in W$. Since $v_a^w = v_a^u$, we have $w \sim_a u$ in M. By the preservation property of f, for every $u' \in f(u)$ we have $u' \sim'_a w'$, so $v_a^{u'} = v_a^{w'}$. Once again, the choice of $w \in W$ does not influence the definition of $\sigma(f)$.

It is easy to check that $\sigma(f)$ is indeed a chromatic simplicial map: preservation of colours is obvious by construction; and for the preservation of simplexes, notice that each facet X_w of $\sigma(M)$ is mapped into the facet $X_{w'}$ of $\sigma(N)$, for some $w' \in f(w)$. However, note that $\sigma(f)(X_w)$ might not in general be a facet; we only know that $\sigma(f)(X_w) \subseteq X_{w'}$.

▶ **Proposition 22.** *σ is functorial, i.e. $\sigma(g \circ f) = \sigma(g) \circ \sigma(f)$.*

Now we can state the main technical result of this paper:

▶ **Theorem 23.** *κ and σ define an equivalence of categories between $\mathsf{KPER}_{\mathsf{A}}^{\mathrm{proper}}$ and $\mathsf{SimCpx}_{\mathsf{A}}$.*

Proof. We have already seen that κ and σ are well-defined functors, it remains to show that:

(i) The composite $\kappa \circ \sigma$ is naturally isomorphic to the identity functor on $\mathsf{KPER}_{\mathsf{A}}^{\mathrm{proper}}$.

(ii) The composite $\sigma \circ \kappa$ is naturally isomorphic to the identity functor on $\mathsf{SimCpx}_{\mathsf{A}}$.

(i) Consider a partial epistemic frame $M = \langle W, \sim \rangle$ in $\mathsf{KPER}_{\mathsf{A}}^{\mathrm{proper}}$. By definition, $\kappa\sigma(M) = \langle F, \sim' \rangle$ where F is the set of facets of $\sigma(M)$. By Lemma 19 there is a bijection $W \cong F$, where a world $w \in W$ if associated with the facet $X_w = \{v_a^w \mid a \in \overline{w}\}$ of $\sigma(M)$. Furthermore, for all $w, w' \in W$, $w \sim_a w'$ iff $X_w \sim'_a X_{w'}$. Indeed, $w \sim_a w' \iff v_a^w = v_a^{w'} \iff a \in \chi(X_w \cap X_{w'})$. Hence, $\kappa\sigma(M)$ and M are isomorphic partial epistemic frames.

Consider a morphism of partial epistemic frames $f : M \to N$, with $M = \langle W, \sim \rangle$ and $N = \langle W', \sim \rangle$. By definition, $\kappa\sigma(f)$ takes a facet X_w of $\sigma(M)$ to a set of facets of $\sigma(N)$, $\kappa\sigma(f)(X_w) = \{Z \in \sigma(N) \mid \sigma(f)(X_w) \subseteq Z\}$. We want to show that this set is equal to $\{X_{w'} \mid w' \in f(w)\}$. Let $w' \in f(w)$. By definition, $\sigma(f)$ maps each vertex v_a^w of X_w to $v_a^{w'}$, so $\sigma(f)(X_w) \subseteq X_{w'}$. Conversely, assume $\sigma(f)(X_w) \subseteq Z$. Since Z is a facet of $\sigma(N)$, $Z = X_{w'}$ for some $w' \in W'$. For each $a \in \overline{w}$, the vertex v_a^w of X_w is mapped by $\sigma(f)$ to $v_a^{x'}$, for $x' \in f(w)$. But since $\sigma(f)(v_a^w) \in Z$, we must have $v_a^{x'} = v_a^{w'}$, so $x' \sim_a w'$. By the saturation property of f, $x' \in f(w)$ implies $w' \in f(w)$ as required. Therefore $\kappa\sigma$ is an isomorphism also on morphisms of partial epistemic frames.

(ii) Consider now a chromatic simplicial complex $\mathcal{C} = \langle V, S, \chi \rangle$. Then $\sigma\kappa(\mathcal{C}) = \langle V', S', \chi' \rangle$ has vertices of the form $V' = \{v_a^Z \mid Z \in \mathcal{F}(\mathcal{C}) \text{ and } a \in \chi(Z)\}$. We must exhibit a bijection $V \cong V'$ which is a chromatic simplicial map in both directions. Given $u \in V$ of colour a, we map it to v_a^Z where Z is any facet of $\mathcal{C}$ that contains u. This is well-defined since any other facet Z' also containing u gives rise to the same vertex $v_a^{Z'} = v_a^Z$, because $Z' \sim_a Z$ in $\kappa(\mathcal{C})$. This map is obviously chromatic, and preserves simplexes because any simplex $Y \in S$ contained in a facet $Z \in \mathcal{F}(\mathcal{C})$ will be mapped to $\{v_a^Z \mid a \in \chi(Y)\} \subseteq X_Z \in \mathcal{F}(\sigma\kappa(\mathcal{C}))$. Conversely, we map a vertex $v_a^Z \in V'$ to the a-coloured vertex of Z. This is also chromatic, and preserves simplexes because any sub-simplex of X_Z is mapped to a sub-simplex of Z. It is easy to check that our two maps form a bijection, therefore $\mathcal{C}$ and $\sigma\kappa(\mathcal{C})$ are isomorphic.

Lastly, consider a chromatic simplicial map $f : \mathcal{C} \to \mathcal{D}$ with $\mathcal{C} = \langle V, S, \chi \rangle$ and $\mathcal{D} = \langle U, R, \zeta \rangle$. As above, we write V' and U' for the vertices of $\sigma\kappa(\mathcal{C})$ and $\sigma\kappa(\mathcal{D})$, respectively. By definition,

$\sigma\kappa(f)$ maps a vertex $v_a^Z \in V'$, with $Z \in \mathcal{F}(C)$, to the vertex $v_a^Y \in U'$, with $Y \in \kappa(f)(Z)$. So by definition of $\kappa(f)$, $f(Z) \subseteq Y$. To prove that $\sigma\kappa(f)$ agrees with f up to the isomorphism of the previous paragraph, we need to show that f sends the a-coloured vertex of Z to the a-coloured vertex of Y. But this is immediate since $f(Z) \subseteq Y$ and f is chromatic. ◀

▶ Remark 24. Note that the equivalence of categories of Theorem 23 strictly extends the one of [10], which was restricted to pure chromatic simplicial complexes on one side and proper epistemic frames on the other. Indeed, if $\mathcal{C}$ is a pure simplicial complex of dimension $|A| - 1$, it is easy to check that $\kappa(\mathcal{C})$ is an epistemic frame, since all agents are alive in all worlds. Moreover, by Proposition 14, the morphisms between those frames are single-valued; so we recover the usual notion of Kripke frame morphism that we had in [10]. Similarly, when M is a proper epistemic frame, the associated simplicial complex $\sigma(M)$ is pure of dimension $|A| - 1$. When restricted to these subcategories, σ and κ are the same functors as in [10].

4 Epistemic logics and their simplicial semantics

Let At be a countable set of atomic propositions and A a finite set of agents. The syntax of epistemic logic formulas $\varphi \in \mathcal{L}_K$ is generated by the following BNF grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \qquad p \in \mathsf{At},\ a \in A$$

We will also use the derived operators, defined as usual: $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \Rightarrow \psi := \neg\varphi \vee \psi$, $\mathsf{true} := p \vee \neg p$, $\mathsf{false} := \neg\mathsf{true}$. Moreover, we assume that the set of atomic propositions is split into a disjoint union of sets, indexed by the agents: $\mathsf{At} = \bigcup_{a\in A} \mathsf{At}_a$. This is usually the case in distributed computing where the atomic propositions represent the local state of a particular agent a. For $U \subseteq A$, we write $\mathsf{At}_U := \bigcup_{a\in U} \mathsf{At}_a$ for the set of atomic propositions concerning the agents in U.

4.1 Partial epistemic models and Simplicial models

In Section 3, we exhibited the equivalence between *partial epistemic frames* and *chromatic simplicial complexes*. In order to give a semantics to epistemic logic, we need to add some extra information on those structures, by labelling the worlds (resp., the facets) with the set of atomic propositions that are true in this world. This gives rise to the notions of *partial epistemic models* and *simplicial models*, respectively. As we shall see, the equivalence of Theorem 23 extends to models in a straightforward manner.

▶ **Definition 25.** *A* partial epistemic model *$M = \langle W, \sim, L\rangle$ over the set of agents A consists of a partial epistemic frame $\langle W, \sim\rangle$ on A, together with function $L : W \to \mathscr{P}(\mathsf{At})$.*

Given another partial epistemic model $M' = \langle W', \sim', L'\rangle$, a morphism *of partial epistemic models $f : M \to M'$ is a morphism of the underlying partial epistemic frames such that for every world $w \in W$ and $w' \in f(w)$, $L'(w') \cap \mathsf{At}_{\overline{w}} = L(w) \cap \mathsf{At}_{\overline{w}}$.*

Let us give some intuition about Definition 25. The set $L(w)$ contains the atomic propositions that are true in the world w. Note that partial epistemic models are simply Kripke models (in the usual sense of modal logics), such that all the accessibility relations $(\sim_a)_{a\in A}$ are PERs. In particular, one might have expected the additional restriction $L(w) \subseteq \mathsf{At}_{\overline{w}}$, saying that a world only contains atomic propositions concerning the alive agents. As we will see in Example 28, there are practical cases where this is not desirable, so we do not impose this. Secondly, recall from Definition 13 that, given a morphism f of partial epistemic frames, a world $w \in W$ and a world $w' \in f(w)$, it is possible that w' has strictly more alive agents than w. When that is the case, in the definition of model morphisms above, we require that the labellings L and L' are preserved only for those agents that are alive in w.

A partial epistemic model is called *proper* when the underlying frame is proper in the sense of Section 3.1. A *pointed partial epistemic model* is a pair (M, w) where w is a world of M. A *morphism* of pointed partial epistemic models $f : (M, w) \to (M', w')$ is a morphism of the partial epistemic models $f : M \to M'$ that preserves the distinguished world, i.e. $w' \in f(w)$. We denote by $\mathcal{PM}_{A,\mathsf{At}}$ (resp. $\mathcal{PM}^*_{A,\mathsf{At}}$) the category of (resp. pointed) proper partial epistemic models over the set of agents A and atomic propositions At.

Recall from Theorem 23 that the worlds of a partial epistemic frame correspond to the facets of the associated chromatic simplicial complex. Thus, to get a corresponding notion of simplicial model, we label the facets by sets of atomic propositions:

▶ **Definition 26.** *A* simplicial model $\mathcal{C} = \langle V, S, \chi, \ell \rangle$ *over the set of agents A consists of a chromatic simplicial complex $\langle V, S, \chi \rangle$ together with a labelling $\ell : \mathcal{F}(\mathcal{C}) \to \mathscr{P}(\mathsf{At})$ that associates with each facet $X \in \mathcal{F}(\mathcal{C})$ a set of atomic propositions.*

Given another simplicial model $\mathcal{D} = \langle V', S', \chi', \ell' \rangle$, a morphism *of simplicial models $f : \mathcal{C} \to \mathcal{D}$ is a chromatic simplicial map such that for all $X \in \mathcal{F}(\mathcal{C})$ and all $Y \in \mathcal{F}(\mathcal{D})$, if $f(X) \subseteq Y$ then $\ell'(Y) \cap \mathsf{At}_{\chi(X)} = \ell(X) \cap \mathsf{At}_{\chi(X)}$.*

A *pointed simplicial model* is a pair $(\mathcal{C}, X)$ where $\mathcal{C}$ is a simplicial model and X is a facet of $\mathcal{C}$. A *morphism* $f : (\mathcal{C}, X) \to (\mathcal{D}, Y)$ of pointed simplicial models is a morphism $f : \mathcal{C} \to \mathcal{D}$ such that $f(X) \subseteq Y$. We denote by $\mathcal{SM}_{A,\mathsf{At}}$ (resp. $\mathcal{SM}^*_{A,\mathsf{At}}$) the category of (resp. pointed) simplicial models over the set of agents A and atomic propositions At. The equivalence of Theorem 23 can be extended to models and pointed models:

▶ **Theorem 27.** *κ and σ induce an equivalence of categories between $\mathcal{SM}_{A,\mathsf{At}}$ (resp. $\mathcal{SM}^*_{A,\mathsf{At}}$) and $\mathcal{PM}_{A,\mathsf{At}}$ (resp. $\mathcal{PM}^*_{A,\mathsf{At}}$).*

▶ **Example 28.** In distributed computing, we are usually interested in reasoning about the input values of the various agents, so the set of atoms is $\mathsf{At} = \{\mathsf{input}_a^x \mid a \in A, x \in \mathsf{Values}\}$. The meaning of the atomic proposition input_a^x is that "agent a has input value x".

Consider again the chromatic simplicial complex $\mathcal{C}$ of Example 6. Here, we have three agents $A = \{a, b, c\}$ and three values $\mathsf{Values} = \{1, 2, 3\}$. Hence, we can construct a simplicial model via the following labelling of facets $\ell : \mathcal{F}(\mathcal{C}) \to \mathscr{P}(\mathsf{At})$.

- For the middle triangle w_1, all three agents are alive and successfully communicated their input values. So, it makes sense to set $\ell(w_1) = \{\mathsf{input}_a^1, \mathsf{input}_b^2, \mathsf{input}_c^3\}$.
- Perhaps more surprisingly, we also choose the same labelling for the six edges adjacent to w_1: $\ell(w_2) = \ell(w_3) = \ell(w_5) = \ell(w_6) = \ell(w_8) = \ell(w_9) = \{\mathsf{input}_a^1, \mathsf{input}_b^2, \mathsf{input}_c^3\}$. Indeed, consider for instance the world w_2, where agent b crashed *after* sending its input value to a. In this world w_2, it is the case that agent a knows that the input of b was 2. Hence, the atomic proposition input_b^2 must be true in w_2, even though the agent b is dead.
- The worlds, w_4, w_7 and w_{10} represent situations where one agent died before being able to send any message. Thus, it is as if only two agents have ever existed, and the labelling only encodes the corresponding two local states: $\ell(w_4) = \{\mathsf{input}_a^1, \mathsf{input}_b^2\}$, $\ell(w_7) = \{\mathsf{input}_b^2, \mathsf{input}_c^3\}$ and $\ell(w_{10}) = \{\mathsf{input}_a^1, \mathsf{input}_c^3\}$.
- Similarly, w_0, w_{11} and w_{12} have labelling $\{\mathsf{input}_a^1\}$, $\{\mathsf{input}_b^2\}$ and $\{\mathsf{input}_c^3\}$ respectively.

We will see in Example 29 some formulas that are true or false in this simplicial model.

4.2 Semantics of epistemic logic

Partial epistemic models are a special case of the usual Kripke models; so we can straightforwardly define the semantics of an epistemic formula $\varphi \in \mathcal{L}_K$ in these models. Formally, gven a pointed partial epistemic model (M, w), we define by induction on φ the *satisfaction relation* $M, w \models \varphi$ which stands for "in the world w of the model M, it holds that φ".

$$\begin{array}{lll} M, w \models p & \text{iff} & p \in L(w) \\ M, w \models \neg\varphi & \text{iff} & M, w \not\models \varphi \\ M, w \models \varphi \wedge \psi & \text{iff} & M, w \models \varphi \text{ and } M, w \models \psi \\ M, w \models K_a\varphi & \text{iff} & M, w' \models \varphi \text{ for all } w' \text{ such that } w \sim_a w' \end{array}$$

We now take advantage of the equivalence with simplicial models (Theorem 27) to define the interpretation of a formula $\varphi \in \mathcal{L}_K(A, P)$ in a simplicial model. Given a pointed simplicial model $(\mathcal{C}, X)$ where $X \in \mathcal{F}(C)$ is a facet of $\mathcal{C}$, we define the relation $\mathcal{C}, X \models \varphi$ by induction:

$$\begin{array}{lll} \mathcal{C}, X \models p & \text{iff} & p \in \ell(X) \\ \mathcal{C}, X \models \neg\varphi & \text{iff} & \mathcal{C}, X \not\models \varphi \\ \mathcal{C}, X \models \varphi \wedge \psi & \text{iff} & \mathcal{C}, X \models \varphi \text{ and } \mathcal{C}, X \models \psi \\ \mathcal{C}, X \models K_a\varphi & \text{iff} & \mathcal{C}, Y \models \varphi \text{ for all } Y \in \mathcal{F}(C) \text{ such that } a \in \chi(X \cap Y) \end{array}$$

▶ **Example 29.** In the simplicial model of Example 28, we have, for instance:

- In world w_1, agent a knows the values of all three agents, i.e. $\mathcal{C}, w_1 \models K_a(\mathsf{input}_a^1 \wedge \mathsf{input}_b^2 \wedge \mathsf{input}_c^3)$ since w_2 and w_3 are indistinguishable from w_1 by agent a and $\mathsf{input}_a^1 \wedge \mathsf{input}_b^2 \wedge \mathsf{input}_c^3$ is true in these three facets. This corresponds to the view of process a, see Example 6.
- In w_3, agent a knows the values of all three agents but agent b only knows the values of a and b: $\mathcal{C}, w_3 \models K_a(\mathsf{input}_a^1 \wedge \mathsf{input}_b^2 \wedge \mathsf{input}_c^3)$ but $\mathcal{C}, w_3 \models K_b(\mathsf{input}_a^1 \wedge \mathsf{input}_b^2)$ and $\mathcal{C}, w_3 \models \neg K_b\,\mathsf{input}_c^3$ since in facet w_4 do not have input_c^3. Similarly, in w_4, agents a and b know each other's values, but do not know the input value of agent c: $\mathcal{C}, w_4 \models K_a(\mathsf{input}_a^1 \wedge \mathsf{input}_b^2)$, $\mathcal{C}, w_4 \models K_b(\mathsf{input}_a^1 \wedge \mathsf{input}_b^2)$, $\mathcal{C}, w_4 \models (\neg K_a\,\mathsf{input}_c^3) \wedge (\neg K_b\,\mathsf{input}_c^3)$
- In world w_1, agent a knows that agent b knows about their respective input values: $\mathcal{C}, w_1 \models K_a K_b(\mathsf{input}_a^1 \wedge \mathsf{input}_b^2)$ but agent a does not know if agent b knows about the value of agent c: $\mathcal{C}, w_1 \models \neg K_a K_b\,\mathsf{input}_c^3$ (because of w_3).

As expected, our two interpretation of $\mathcal{L}_K$ agree up to the equivalence of Theorem 27:

▶ **Proposition 30.** *Given a pointed simplicial model* $(\mathcal{C}, X)$, $\mathcal{C}, X \models \varphi$ *iff* $\kappa(\mathcal{C}, X) \models \varphi$. *Conversely, given a pointed proper partial epistemic model* (M, w), $M, w \models \varphi$ *iff* $\sigma(M, w) \models \varphi$.

This is straightforward by induction on the structure of the formula φ.

4.3 Reasoning about alive and dead agents

In Example 29, we only considered formulas talking about what the agents know about each other's input values. It is a natural idea to also contemplate formulas expressing which agents are alive or dead, for example "agent a knows that agent b is dead". Fortunately, such formulas can already be expressed in our logic without any extra work, as derived operators $\mathsf{dead}(a) := K_a\,\mathsf{false}$, and $\mathsf{alive}(a) := \neg\mathsf{dead}(a)$. It is easy to check that indeed:

- In partial epistemic models, $M, w \models \mathsf{alive}(a)$ iff $w \sim_a w$.
- In simplicial models, $\mathcal{C}, X \models \mathsf{alive}(a)$ iff $a \in \chi(X)$.

▶ **Example 31.** Consider again the simplicial model of Examples 6 and 28, and its corresponding partial epistemic model of Example 10. It is easy to see that:

- $M, w_3 \models \mathsf{alive}(b) \wedge \mathsf{alive}(a)$ but $M, w_3 \models \mathsf{dead}(c)$,
- $M, w_1 \models \neg K_a\,\mathsf{alive}(c)$ since e.g. $M, w_3 \models \mathsf{dead}(c)$ whereas $M, w_1 \models \mathsf{alive}(c)$,

- Agents a and b know, in world w_4, that c is dead: $M, w_4 \models K_b\,\mathsf{dead}(c) \wedge K_a\,\mathsf{dead}(c)$ since, first, in world w_3 (which is indistinguishable from w_3 by agent b), agent c is not alive, and second, in world w_5 (which is indistinguishable from w_3 by agent a, c is not alive either.

In w_4 everything looks as if agents a and b were executing solo, without c ever existing, whereas in worlds w_3 and w_5, agent c dies at some point, but has been active and its local value has been observed by one of the other agents.

4.4 The axiom system $\mathbf{KB4_n}$

We consider the usual proof theory of normal modal logics, with all propositional tautologies, closure by modus ponens, and the necessitation rule: if φ is a tautology, then $K_a\varphi$ is a tautology. In normal modal logics, there is a well-known correspondence between properties of Kripke models that we consider, and corresponding axioms that make the logic sound and complete [8]. In our case, partial epistemic models are symmetric and transitive. Thus we get the logic $\mathbf{KB4_n}$, obeying the following additional axioms.

$$\mathbf{K} : K_a(\varphi \Rightarrow \psi) \implies (K_a\varphi \Rightarrow K_a\psi)$$
$$\mathbf{B} : \varphi \implies K_a\neg K_a\neg\varphi$$
$$\mathbf{4} : K_a\varphi \implies K_aK_a\varphi$$

The difference between $\mathbf{KB4_n}$ and the more standard multi-agent epistemic logics $\mathbf{S5_n}$ is that we do not necessarily have axiom $\mathbf{T}$: $K_a\varphi \implies \varphi$. Axiom $\mathbf{T}$ is valid in Kripke models whose accessibility relation is reflexive, which we do not enforce. The logic $\mathbf{KB4_n}$ is in fact equivalent to $\mathbf{KB45_n}$ (see e.g. [8]), so we also have for free the Axiom $\mathbf{5}$, which corresponds to Euclidean Kripke frames. We have the following well-known result, see e.g. [6].

▶ **Theorem 32.** *The axiom system* $\mathbf{KB4_n}$ *is sound and complete with respect to the class of partial epistemic models.*

Here are a few examples of valid formulas in $\mathbf{KB4_n}$, related to the liveness of agents.

- Dead agents know everything: $\mathbf{KB4_n} \vdash \mathsf{dead}(a) \implies K_a\varphi$.
- Alive agents know they are alive: $\mathbf{KB4_n} \vdash \mathsf{alive}(a) \implies K_a\,\mathsf{alive}(a)$.
- Alive agents satisfy Axiom $\mathbf{T}$: $\mathbf{KB4_n} \vdash \mathsf{alive}(a) \implies (K_a\varphi \Rightarrow \varphi)$.
- Only alive agents matter for $K_a\varphi$: $\mathbf{KB4_n} \vdash K_a\varphi \iff (\mathsf{alive}(a) \Rightarrow K_a\varphi)$.

As an application of the fourth tautology, notice that a formula of the form $K_aK_b\varphi$ is equivalent to $K_a(\mathsf{alive}(b) \Rightarrow K_b\varphi)$. So, to check whether this formula is true in some pointed model (M, w), we only need to check that $K_b\varphi$ is true in the worlds $w' \sim_a w$ where b is alive.

4.5 Completeness for simplicial models

According to Theorem 27, simplicial models are equivalent to *proper* partial epistemic models. Thus Theorem 32 does not apply directly to simplicial models, and some extra care must be taken to deal with this "proper" requirement. Indeed, it is easy to check that the two formulas below are true in every simplicial model; but they are not provable in $\mathbf{KB4_n}$.

$$\mathbf{NE}\text{:} \quad \bigvee_{a\in A} \mathsf{alive}(a)$$
$$\mathbf{SA_a}\text{:} \quad \Big(\mathsf{alive}(a) \wedge \bigwedge_{b\neq a} \mathsf{dead}(b)\Big) \implies K_a \bigwedge_{b\neq a} \mathsf{dead}(b)$$

The formula $\mathbf{NE}$ (Non-Emptiness) says that in every world, there is at least one agent that is alive; and the formula $\mathbf{SA_a}$ (Single Agent) says that if there is exactly one alive agent a, this agent knows that all other agents are dead. It is straightforward to check that:

▶ **Proposition 33.** *The axiom system* $\mathbf{KB4_n} + \mathbf{NE} + (\mathbf{SA_a})_{a \in A}$ *is sound with respect to the class of simplicial models.*

We also believe that this axiom system is complete; but the proof is more involved and we leave it for future work. A proof sketch is given in the Appendix G. The axioms **NE** and $\mathbf{SA_a}$ embody the "hidden" assumptions in the use of simplicial models. Note that we could easily get rid of **NE** by allowing the existence of a fictitious (-1)-dimensional simplex representing an empty world. This is known in geometry as *augmented* simplicial complexes. However, the axioms $\mathbf{SA_a}$ are more substantial.

▶ **Conjecture 34.** $\mathbf{KB4_n} + \mathbf{NE} + (\mathbf{SA_a})_{a \in A}$ *is complete w.r.t. the class of simplicial models.*

4.6 Knowledge gain

In [10], a key property of the logic used in distributed computing applications is the so-called "knowledge gain" property. This principle says that agents cannot acquire new knowledge along morphisms of simplicial models. Namely, what is known in the image of a morphism was already known in the domain. The knowledge gain property is used when we want to prove that a certain simplicial map $f : \mathcal{C} \to \mathcal{D}$ cannot exist. To achieve this, we choose a formula φ and show that φ is true in every world of $\mathcal{D}$, and that φ is false in at least one world of $\mathcal{C}$. Then by the knowledge gain property, the map f does not exist. Such a formula φ is called a *logical obstruction.* While we are not interested in proving distributed computing results in this paper (the synchronous crash model of Figure 2 is merely an illustrative example), we still check that some version of the knowledge gain property holds, as a sanity check towards future work.

The knowledge gain property that appeared in [10] applied to *positive* epistemic formulas, i.e., they are cannot talk about what an agent does not know. Here, we also require an additional condition, which says that every atomic proposition $p \in \mathsf{At}_a$ that appears in the formula must be *guarded* by a conditional making sure that agent a is alive. This is because there might be agents that are dead in the domain of a morphism, but are alive in the codomain.

Formally, the fragment of *guarded positive epistemic formulas* $\varphi \in \mathcal{L}^+_{K,\text{alive}}$ is defined by the grammar $\varphi ::= \mathsf{alive}(B) \Rightarrow \psi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid K_a\varphi,\ \ a \in A,\ \ B \subseteq A,\ \ \psi \in \mathcal{L}\restriction_B$, where the formula $\mathsf{alive}(B)$ stands for $\bigwedge_{a \in B} \mathsf{alive}(a)$, and the formula $\psi \in \mathcal{L}\restriction_B$ is a *propositional formula restricted to the agents in* B, defined formally by the grammar: $\psi ::= p \mid \neg\psi \mid \psi \wedge \psi,\ \ p \in \mathsf{At}_B$.

▶ **Theorem 35** (knowledge gain)**.** *Consider simplicial models* $\mathcal{C} = \langle V, S, \chi, \ell \rangle$ *and* $\mathcal{D} = \langle V', S', \chi', \ell' \rangle$*, and a morphism of pointed simplicial models* $f : (\mathcal{C}, X) \to (\mathcal{D}, Y)$*. Let* $\varphi \in \mathcal{L}^+_{K,\text{alive}}$ *be a guarded positive epistemic formula. Then* $\mathcal{D}, Y \models \varphi$ *implies* $\mathcal{C}, X \models \varphi$*.*

5 Conclusion

We began exposing the interplay between epistemic logics and combinatorial geometry in [10]. The importance of this perspective has been well established in distributed computing, where the topology of the simplicial model determines the solvability of a distributed task [12]. Here we extended it to situations where agents may die: impure simplicial complexes need to be considered. Many technical interesting issues arise, which shed light on the epistemic assumptions hiding behind the use of simplicial models.

But the main point is that our work opens the way to give a formal epistemic semantics to distributed systems where processes may fail and failures are detectable (as in the synchronous crash failure model). It would be interesting to use our simplicial model to reason about the solvability of tasks in such systems, for example, the following have not been studied using epistemic logic, to the best of our knowledge: non-complete communication (instead of broadcast situation we considered here) graphs [3], and tasks such as renaming [21] and lattice agreement [28]. Especially interesting would be extending the set agreement logical obstruction of [27] to the synchronous crash setting.

Finally, we hope that our simplicial semantics can be useful to reason not only about distributed computing, but also about in other situations with interactions beyond pairs of agents [1]. For instance, impure simplicial complexes have been shown to occur when modelling social systems, neuroscience, and other biological systems (see e.g. [19]).

References

1 Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92, 2020. Networks beyond pairwise interactions: Structure and dynamics. doi:10.1016/j.physrep.2020.05.004.

2 Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 91–100. ACM, 1993. doi:10.1145/167088.167119.

3 Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. Synchronous t-resilient consensus in arbitrary graphs. In Mohsen Ghaffari, Mikhail Nesterenko, Sébastien Tixeuil, Sara Tucci, and Yukiko Yamauchi, editors, *Stabilization, Safety, and Security of Distributed Systems - 21st International Symposium, SSS 2019, Pisa, Italy, October 22-25, 2019, Proceedings*, volume 11914 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2019. doi:10.1007/978-3-030-34992-9_5.

4 Hans van Ditmarsch, Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. Knowledge and simplicial complexes. *CoRR*, abs/2002.08863, 2020. To appear in Philosophy of Computing-Themes from IACAP 2019, Eds.: Björn Lundgren, and Nancy Abigail Nuñez Hernández. arXiv:2002.08863.

5 Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Inf. Comput.*, 88(2):156–186, 1990. doi:10.1016/0890-5401(90)90014-9.

6 Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 2003.

7 Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982. doi:10.1016/0020-0190(82)90033-3.

8 James Garson. Modal Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2021 edition, 2021.

9 Éric Goubault, Marijana Lazic, Jérémy Ledent, and Sergio Rajsbaum. Wait-free solvability of equality negation tasks. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, volume 146 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.DISC.2019.21.

10 Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. A simplicial complex model for dynamic epistemic logic to study distributed task computability. *Inf. Comput.*, 278:104597, 2021. doi:10.1016/j.ic.2020.104597.

11 Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *J. ACM*, 37(3):549–587, 1990. doi:10.1145/79147.79161.

12 M. Herlihy, D. Kozlov, and S. Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, San Francisco, CA, USA, 2013.

13 M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, November 1999. doi:10.1145/331524.331529.

14 Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, January 1991. doi:10.1145/114005.102808.

15 Maurice Herlihy, Sergio Rajsbaum, and Mark R. Tuttle. An overview of synchronous message-passing and topology. *Electronic Notes in Theoretical Computer Science*, 39(2):1–17, 2000. doi:10.1016/S1571-0661(05)01148-5.

16 Dmitry N. Kozlov. *Combinatorial Algebraic Topology*, volume 21 of *Algorithms and computation in mathematics*. Springer, 2008. doi:10.1007/978-3-540-71962-5.

17 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

18 John C. Mitchell and Eugenio Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, 51:99–124, 1996.

19 Andrea Mock and Ismar Volic. Political structures and the topology of simplicial complexes, 2021.

20 Yoram Moses. *Knowledge in Distributed Systems*, pages 1051–1055. Springer New York, New York, NY, 2016. doi:10.1007/978-1-4939-2864-4_606.

21 Michael Okun. Strong order-preserving renaming in the synchronous message passing model. *Theor. Comput. Sci.*, 411(40–42):3787–3794, September 2010. doi:10.1016/j.tcs.2010.06.001.

22 Y. Moses R. Fagin, J. Halpern and M. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

23 Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000. doi:10.1137/S0097539796307698.

24 Hans van Ditmarsch. Wanted dead or alive: Epistemic logic for impure simplicial complexes. In Alexandra Silva, Renata Wassermann, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information, and Computation - 27th International Workshop, WoLLIC 2021, Virtual Event, October 5-8, 2021, Proceedings*, volume 13038 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2021. doi:10.1007/978-3-030-88853-4_3.

25 Hans van Ditmarsch, Éric Goubault, Marijana Lazic, Jérémy Ledent, and Sergio Rajsbaum. A dynamic epistemic logic analysis of equality negation and other epistemic covering tasks. *J. Log. Algebraic Methods Program.*, 121:100662, 2021. doi:10.1016/j.jlamp.2021.100662.

26 Frans Voorbraak. Generalized kripke models for epistemic logic. In *Proceedings of the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge*, TARK '92, pages 214–228, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

27 Koki Yagi and Susumu Nishimura. Logical obstruction to set agreement tasks for superset-closed adversaries, 2021.

28 Xiong Zheng and Vijay K. Garg. Byzantine lattice agreement in synchronous message passing systems. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 32:1–32:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.32.

A Proof of Proposition 17

We break down Proposition 17 into three statements:

▶ **Proposition 36.** *$\kappa(\mathcal{C})$ is a proper partial epistemic frame.*

Proof. The relation $\sim_a$ on facets is easily seen to be symmetric and transitive, because there can be at most one vertex $v \in X \cap Y$ with $\chi(v) = a$. To show that $\kappa(\mathcal{C})$ is proper, consider two worlds X and Y in $\kappa(\mathcal{C})$, i.e., two facets of $\mathcal{C}$. In simplicial complexes, $X \neq Y$ implies that at least one vertex of X, say v, does not belong to Y: otherwise, we would have $X \subseteq Y$ so X would not be a facet. Let $a = \chi(v)$ be the colour of v. Then a is alive in X because $a \in \chi(X \cap X)$; and $X \not\sim_a Y$ because $v \notin X \cap Y$ and there can be only one vertex with colour a in X. ◀

▶ **Proposition 37.** *$\kappa(f)$ is a morphism of partial epistemic frames from $\kappa(\mathcal{C})$ to $\kappa(\mathcal{D})$.*

Proof. Assume X and Y are facets of $\mathcal{C} = \langle V, S, \chi\rangle$ such that $X \sim_a Y$ in $\kappa(\mathcal{C})$. So there is a vertex $v \in V$ such that $v \in X \cap Y$ and $\chi(v) = a$. Therefore $f(v)$ is in all facets $Z \in \kappa(\mathcal{D})$ such that $f(X) \subseteq Z$ and all facets $T \in \kappa(\mathcal{D})$ such that $f(Y) \subseteq T$. As $\chi(f(v)) = a$, this means that $a \in \chi(Z \cap T)$, hence, for all $Z \in \kappa(f)(X)$ and $T \in \kappa(f)(Y)$, $Z \sim_a T$. Furthermore, $\kappa(f)(X)$ as defined is obviously saturated, so $\kappa(f)$ is a morphism of partial epistemic frames. ◀

▶ **Proposition 38.** *κ is functorial, i.e. $\kappa(g \circ f) = \kappa(g) \circ \kappa(f)$.*

Proof. Let $f : \mathcal{C} \to \mathcal{D}$ and $g : \mathcal{D} \to \mathcal{E}$ be two chromatic simplicial maps. By definition, for a world/facet $X \in \kappa(\mathcal{C})$, we have $\kappa(g \circ f)(X) = \{Z' \in \mathcal{F}(\mathcal{E}) \mid (g \circ f)(X) \subseteq Z'\}$, while $(\kappa(g) \circ \kappa(f))(X) = \mathsf{sat}_{\chi(X)}(Z)$ for some facets $Z \in \kappa(g)(Y)$ and $Y \in \kappa(f)(X)$. We show that they are equal.

Consider Z' such that $(g \circ f)(X) \subseteq Z'$; we need to show that $Z' \sim_a Z$ for all $a \in \chi(X)$. Indeed, let v be the a-coloured vertex of X. Then $(g \circ f)(v) \in Z'$ by assumption, and $(g \circ f)(v) \in Z$ because $f(v) \in Y$. So there is an a-coloured vertex $(g \circ f)(v) \in Z' \cap Z$.

Conversely, let $Z' \in \mathsf{sat}_{\chi(X)}(Z)$, i.e. $Z' \sim_a Z$ for all $a \in \chi(X)$. Let v be a vertex of X, and let $a = \chi(v)$. Since $f(v) \in Y$, we have $(g \circ f)(v) \in Z$. Since Z can have only one a-colored vertex and $a \in \chi(Z' \cap Z)$, we get $(g \circ f)(v) \in Z'$. Thus $(g \circ f)(X) \subseteq Z'$ as required. ◀

B Proof of Proposition 19

Proof. Each world $w \in W$ is associated with the simplex $X_w = \{v_a^w \mid a \in \overline{w}\}$. We need to prove that these simplexes are indeed facets, and that they are distinct for $w \neq w'$. It suffices to show that for all $w \neq w'$, $X_w \not\subseteq X_{w'}$. Since M is proper, there exists an agent a which is alive in w such that $w \not\sim_a w'$. Then, either a is alive in w', in which case $v_a^w \neq v_a^{w'}$, or a is dead in w'. In both cases, v_a^w is not a vertex of $X_{w'}$ so $X_w \not\subseteq X_{w'}$. ◀

C Proof of Proposition 22

Proof. Let $f : M \to N$ and $g : N \to P$ be morphisms of partial epistemic frames. Let v_a^w be a vertex of $\sigma(M)$, where $w \in W$ is a world of M. By definition, $\sigma(g \circ f)(v_a^w) = v_a^{w''}$ where $w'' \in (g \circ f)(w)$; whereas $(\sigma(g) \circ \sigma(f))(v_a^w) = v_a^{y''}$ where $y'' \in g(y')$ and $y' \in f(w)$. To show that they are the same vertex, we need to prove that $w'' \sim_a y''$. By definition of $(g \circ f)(w)$, there exists $x' \in f(w)$ and $x'' \in g(x')$ such that $w'' \sim_a x''$. Since $w \sim_a w$, we have $x' \sim_a y'$ by the preservation property of f, and then $x'' \sim_a y''$ again by preservation. Finally, $w'' \sim_a y''$ by transitivity. ◀

D Proof of Theorem 27

Proof. For a simplicial model $\mathcal{C} = \langle V, S, \chi, \ell\rangle$, recall that the worlds of the associated partial epistemic frame are the facets of $\mathcal{C}$; so the labelling in $\kappa(\mathcal{C})$ is $L(X) = \ell(X)$ for $X \in \mathcal{F}(\mathcal{C})$. For a partial epistemic model $M = \langle W, \sim, L\rangle$, recall that the facets of the associated chromatic simplicial complex are of the form X_w for $w \in W$; so to define $\sigma(M)$, we set $\ell(X_w) = L(w)$. For the pointed version, we similarly define $\kappa(\mathcal{C}, X) = (\kappa(\mathcal{C}), X)$ and $\sigma(M, w) = (\sigma(M), X_w)$.

Checking that this is indeed an equivalence of category is an immediate consequence of Theorem 23. The only detail to check is that the extra conditions on morphisms are preserved: if f is a morphism of (pointed) simplicial models, then $\kappa(f)$ is a morphism of (pointed) partial epistemic models. Indeed, $f(X) \subseteq Y$ implies that $Y \in \kappa(f)(X)$ by definition of $\kappa(f)$. Similarly, if g is a morphism of (pointed) partial epistemic models, then $\sigma(g)$ is a morphism of (pointed) simplicial models. ◀

E Proofs of the sample valid formulas in $\mathrm{KB4_n}$, Section 4.4

Proof. We begin by proving that $\mathbf{KB4_n} \vdash \mathsf{dead}(a) \Rightarrow K_a\varphi$. By the **K** axiom, we have $K_a(\mathsf{false} \Rightarrow \varphi) \Rightarrow (K_a\mathsf{false} \Rightarrow K_a\varphi)$. But $\mathsf{false} \Rightarrow \varphi$ is a tautology, and by the necessitation rule, $K_a(\mathsf{false} \Rightarrow \varphi)$ is a tautology. Hence $K_a\mathsf{false} \Rightarrow K_a\varphi$ but $\mathsf{dead}(a) \equiv K_a\mathsf{false}$.

We then prove that $\mathbf{KB4_n} \vdash \mathsf{alive}(a) \Rightarrow K_a\,\mathsf{alive}(a)$. By axiom **B** we know that $\mathsf{true} \Rightarrow K_a\neg K_a\mathsf{false}$, that is, $\mathsf{true} \Rightarrow K_a\mathsf{alive}(a)$, hence $K_a\mathsf{alive}(a)$. As a matter of fact, either a is dead and it knows everything by the first property above, even $K_a\mathsf{alive}(a)$ or a is alive, and knows it is alive.

Now we prove that $\mathbf{KB4_n} \vdash \mathsf{alive}(a) \Rightarrow (K_a\varphi \Rightarrow \varphi)$. We will show the contrapositive, $\mathbf{KB4_n} \vdash (K_a\,\varphi \wedge \neg\varphi) \Rightarrow \mathsf{dead}(a)$. Assume $K_a\,\varphi$ and $\neg\varphi$, we want to show $\mathsf{dead}(a)$, i.e. $K_a\,\mathsf{false}$. By axiom **B**, $\neg\varphi \Rightarrow K_a\neg K_a\,\varphi$; so by modus ponens, $K_a\neg K_a\,\varphi$. Moreover, by axiom **4** and the assumption of $K_a\,\varphi$, we get $K_aK_a\,\varphi$. Therefore, since we proved both $K_a\neg K_a\,\varphi$ and $K_aK_a\,\varphi$, by axiom **K** and modus ponens, we obtain $K_a\mathsf{false}$.

Finally we prove that $\mathbf{KB4_n} \vdash K_a\varphi \iff (\mathsf{alive}(a) \Rightarrow K_a\varphi)$. The left to right implication is trivial. Now suppose $\mathsf{alive}(a) \Rightarrow K_a\varphi$, we want to prove that $K_a\varphi$. By modus ponens $\mathsf{dead}(a) \vee \mathsf{alive}(a)$ and if $\mathsf{dead}(a)$ then a knows everything by the first property we proved, for instance $K_a\varphi$. If $\mathsf{alive}(a)$ then, because $\mathsf{alive}(a) \Rightarrow K_a\varphi$, $K_a\varphi$ holds. ◀

F Proof of Proposition 33

Proof. Let us first consider axiom **NE**: $\bigvee_{a\in A}\mathsf{alive}(a)$. Take a proper epistemic model $M = \langle W, \sim\rangle$. To prove that for all $w \in W$, $M, w \models \mathbf{NE}$, we have to prove that there exists $a \in A$ such that $w \sim_a w$. This is by definition of properness.

We now turn to axiom $\mathbf{SA_a}$: $\left(\mathsf{alive}(a) \wedge \bigwedge_{b\neq a}\mathsf{dead}(b)\right) \implies K_a\bigwedge_{b\neq a}\mathsf{dead}(b)$. Take M again, a proper epistemic frame, and $w \in W$ such that $M, w \models \mathsf{alive}(a) \wedge \bigwedge_{b\neq a}\mathsf{dead}(b)$. We must prove that $M, w \models K_a\bigwedge_{b\neq a}\mathsf{dead}(b)$. As $M, w \models \mathsf{alive}(a) \wedge \bigwedge_{b\neq a}\mathsf{dead}(b)$, $w \sim_a w$ and, for all $b \neq a$, there is no w' such that $w' \sim_b w$.

Consider now any u such that $u \sim_a w$, we need to show that for all $b \neq a$, $M, u \models \mathsf{dead}(b)$, i.e. that $u \not\sim_b u$. But in w, only a is alive, and by the properness property of M, such a u is necessarily equal to w. This is because if $u \neq w$, it has to be distinguished by some agent that is alive in w, which can only be a by hypothesis on w, which contradicts the fact that $u \sim_a w$. Therefore we have trivially $M, u \models \mathsf{dead}(b)$ since $M, w \models \mathsf{dead}(b)$. ◀

G Proof sketch of Conjecture 34

We prove completeness for the class of proper partial epistemic models. Completeness for simplicial models then follows directly by Proposition 30.

▶ **Lemma 39.** *The axiom system* $\mathbf{KB4_n} + \mathbf{NE} + (\mathbf{SA_a})_{a\in A}$ *is complete w.r.t. the class of proper partial epistemic models.*

Proof sketch. As usual in completeness proofs, we build a canonical model M^c whose worlds are maximal and consistent sets of formulas (for the logic $\mathbf{KB4_n} + \mathbf{NE} + (\mathbf{SA_a})_{a \in A}$). The usual machinery (Lindenbaum's Lemma, the Truth Lemma) works as expected.

All we have to do to complete the proof is show that M^c is a proper partial epistemic model. Showing that M^c is a partial epistemic model is standard (see e.g. [8]): the axioms **B** and **4** are used to prove symmetry and transitivity, respectively. However, the model M^c is in fact not proper: while the axiom **NE** ensures that every world has at least one alive agent, non-proper behaviour (such as the one of Example 9) can occur within M^c.

To fix this, we resort to the classic *unwinding* construction. From M^c, we build an unwinded model $U(M^c)$ whose worlds are paths in M^c, of the form $(w_0, a_1, w_1, \ldots, a_k, w_k)$, where each w_i is a world of M^c and for all i, $w_i \sim_{a_{i+1}} w_{i+1}$. This model $U(M^c)$ can be shown to be bisimilar to M^c. Moreover, $U(M^c)$ is proper: behaviours such as the one of Example 9 are ruled out by the unwinding construction. The only remaining possibility for non-properness concerns worlds with a unique agent; they are ruled out by the axioms $\mathbf{SA_a}$. ◀

H Proof of Theorem 35

Proof. We proceed by induction on the structure of the guarded positive formula φ.

For the base case, assume $\varphi = \mathsf{alive}(B) \Rightarrow \psi$ for some set of agents $B \subseteq A$ and some propositional formula $\psi \in \mathcal{L}\restriction_B$. We distinguish two cases. Either some agent $a \in B$ is dead in the world X, in which case $\mathcal{C}, X \models \varphi$ is true. Or all agents in B are alive in X, and since $f(X) \subseteq Y$ (because f is a morphism of pointed simplicial models), all agents in B are also alive in Y. Thus, we have $\mathcal{D}, Y \models \psi$. Moreover, since f is a morphism, we know that $\ell(X) \cap \mathsf{At}_{\chi(X)} = \ell(Y) \cap \mathsf{At}_{\chi(X)}$. In particular, this yields $\ell(X) \cap \mathsf{At}_B = \ell(Y) \cap \mathsf{At}_B$ because $B \subseteq \chi(X)$. So all atomic propositions in At_B have the truth value in the worlds X and Y. As a consequence $\mathcal{D}, Y \models \psi$ implies that $\mathcal{C}, X \models \psi$, and thus $\mathcal{C}, X \models \varphi$ as required.

The cases of conjunction and disjunction follow trivially from the induction hypothesis. Finally, for the case of a formula $K_a\varphi$, suppose that $\mathcal{D}, Y \models K_a\varphi$. If $a \notin \chi(X)$ then $\mathcal{C}, X \models K_a\varphi$, trivially (dead agents know everything). So let us assume that $a \in \chi(X)$. In order to show $\mathcal{C}, X \models K_a\varphi$, assume that $a \in \chi(X \cap X')$ for some facet X', and let us prove $\mathcal{C}, X' \models \varphi$. Let v be the a-coloured vertex in $X \cap X'$. Then $f(v) \in f(X) \cap f(X')$. Recall that $f(X) \subseteq Y$ by assumption, and let Y' be a facet of $\mathcal{D}$ containing $f(X')$. So $f(v) \in Y \cap Y'$, and since $\chi(f(v)) = a$, we get $a \in \chi(Y \cap Y')$ and thus $\mathcal{D}, Y' \models \varphi$. By induction hypothesis, we obtain $\mathcal{C}, X' \models \varphi$. ◀

Star Transposition Gray Codes for Multiset Permutations

Petr Gregor
Department of Theoretical Computer Science and Mathematical Logic,
Charles University, Prague, Czech Republic

Torsten Mütze
Department of Computer Science, University of Warwick, Coventry, UK
Department of Theoretical Computer Science and Mathematical Logic,
Charles University, Prague, Czech Republic

Arturo Merino
Department of Mathematics, TU Berlin, Germany

Abstract

Given integers $k \geq 2$ and $a_1, \ldots, a_k \geq 1$, let $\boldsymbol{a} := (a_1, \ldots, a_k)$ and $n := a_1 + \cdots + a_k$. An $\boldsymbol{a}$-multiset permutation is a string of length n that contains exactly a_i symbols i for each $i = 1, \ldots, k$. In this work we consider the problem of exhaustively generating all $\boldsymbol{a}$-multiset permutations by star transpositions, i.e., in each step, the first entry of the string is transposed with any other entry distinct from the first one. This is a far-ranging generalization of several known results. For example, it is known that permutations ($a_1 = \cdots = a_k = 1$) can be generated by star transpositions, while combinations ($k = 2$) can be generated by these operations if and only if they are balanced ($a_1 = a_2$), with the positive case following from the middle levels theorem. To understand the problem in general, we introduce a parameter $\Delta(\boldsymbol{a}) := n - 2\max\{a_1, \ldots, a_k\}$ that allows us to distinguish three different regimes for this problem. We show that if $\Delta(\boldsymbol{a}) < 0$, then a star transposition Gray code for $\boldsymbol{a}$-multiset permutations does not exist. We also construct such Gray codes for the case $\Delta(\boldsymbol{a}) > 0$, assuming that they exist for the case $\Delta(\boldsymbol{a}) = 0$. For the case $\Delta(\boldsymbol{a}) = 0$ we present some partial positive results. Our proofs establish Hamilton-connectedness or Hamilton-laceability of the underlying flip graphs, and they answer several cases of a recent conjecture of Shen and Williams. In particular, we prove that the middle levels graph is Hamilton-laceable.

2012 ACM Subject Classification Mathematics of computing → Permutations and combinations; Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Matchings and factors

Keywords and phrases Gray code, permutation, combination, transposition, Hamilton cycle

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.34

Related Version Full proofs are provided in the preprint version [12] of this extended abstract.
Full Version: `https://arxiv.org/abs/2108.07465`

Funding This work was supported by Czech Science Foundation grant GA 19-08554S, and by German Science Foundation grant 413902284.
Arturo Merino: was also supported by ANID Becas Chile 2019-72200522.

Acknowledgements We thank Jiří Fink for inspiring brain-storming sessions in the early phases of this research project, and we thank Aaron Williams for interesting discussions about star transposition Gray codes.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 34; pp. 34:1–34:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Permutations and combinations are two of the most fundamental classes of combinatorial objects. Specifically, *k-permutations* are all linear orderings of $[k] := \{1, \ldots, k\}$, and their number is $k!$. Moreover, *(α, β)-combinations* are all β-element subsets of $[n]$ where $n := \alpha + \beta$, and their number is $\binom{n}{\alpha} = \binom{n}{\beta}$. Permutations and combinations are generalized by so-called multiset permutations, and in this paper we consider the task of listing them such that any two consecutive objects in the list differ by particular transpositions, i.e., by swapping two elements. Such a listing of objects subject to a "small change" operation is often referred to as *Gray code* [27, 30]. One of the standard references for algorithms that efficiently generate various combinatorial objects, including permutations and combinations, is Knuth's book [19] (see also [25]).

1.1 Permutation generation

There is a vast number of Gray codes for permutation generation, most prominently the Steinhaus-Johnson-Trotter algorithm [18, 40], which generates all k-permutations by adjacent transpositions, i.e., swaps of two neighboring entries of the permutation; see Figure 1 (a). In this work, we focus on *star transpositions*, i.e., swaps of the first entry of the permutation with any later entry. An efficient algorithm for generating permutations by star transpositions was found by Ehrlich, and it is described as Algorithm E in Knuth's book [19, Section 7.2.1.2]; see Figure 1 (b). For any permutation generation algorithm based on transpositions, we can define the *transposition graph* as the graph with vertex set $[k]$, and an edge between i and j if the algorithm uses transpositions between the ith and jth entry of the permutation. Clearly, the transposition graph for adjacent transpositions is a path, whereas the transposition graph for star transpositions is a star (hence the name "star transposition"). In fact, Kompel'maher and Liskovec [20], and independently Slater [36], showed that all k-permutations can be generated for any transposition tree on $[k]$. Transposition Gray codes for permutations with additional restrictions were studied by Compton and Williamson [7] and by Shen and Williams [33].

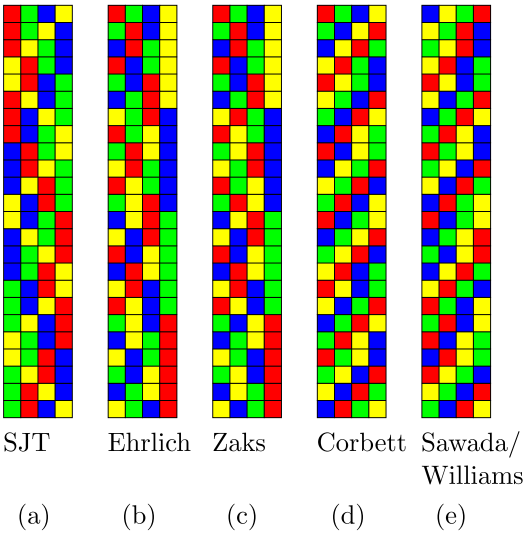


Figure 1 Gray codes for 4-permutations (SJT=Steinhaus-Johnson-Trotter).

Several known algorithms for permutation generation use operations other than transpositions. Specifically, Zaks [43] presented an algorithm for generating permutations by prefix reversals; see Figure 1 (c). Moreover, Corbett [8] showed that all k-permutations can be generated by cyclic left shifts of any prefix of the permutation by one position; see Figure 1 (d). Another notable result is Sawada and Williams' recent solution [32] of the Sigma-Tau problem, proving that all k-permutations can be generated by cyclic left shifts of the entire permutation by one position or transpositions of the first two elements; see Figure 1 (e).

All of the aforementioned results can be seen as explicit constructions of Hamilton paths in the Cayley graph of the symmetric group, generated by different sets of generators (transpositions, reversals, or shifts). It is an open problem whether the Cayley graph of the symmetric group has a Hamilton path for any set of generators [28]. This is a special case of the well-known open problem whether any connected Cayley graph has a Hamilton path, or even more generally, whether this is the case for any vertex-transitive graph [21].

1.2 Combination generation and the middle levels conjecture

In a computer, (α,β)-combinations can be conveniently represented by bitstrings of length $n := \alpha+\beta$, where the ith bit is 1 if the element i is in the set and 0 otherwise. For example, the $(5,3)$-combination $\{1,6,7\}$ is represented by the string 10000110.

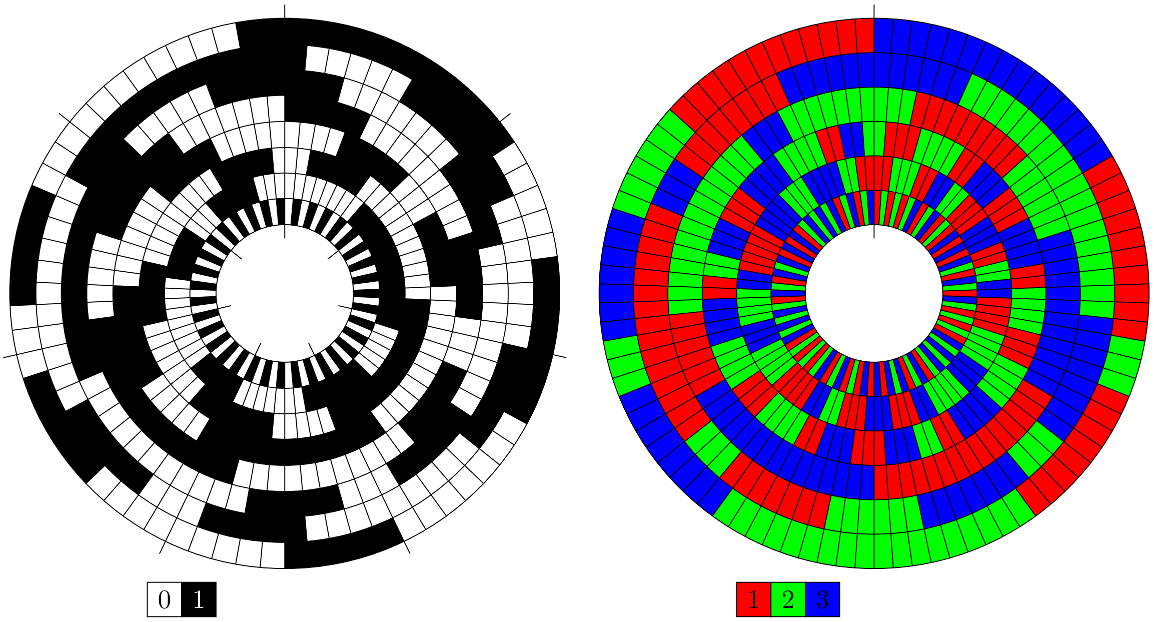
In the 1980s, Buck and Wiedemann [3] conjectured that all (α,α)-combinations can be generated by star transpositions for every $\alpha \geq 1$, i.e., in every step we swap the first bit of the bitstring representation with a later bit. Figure 2 (a) shows such a star transposition Gray code for $(4,4)$-combinations. Buck and Wiedemann's conjecture was raised independently by Havel [15], as a question about the existence of a Hamilton cycle through the middle two levels of the $(2\alpha-1)$-dimensional hypercube. This conjecture became known as *middle levels conjecture*, and it attracted considerable attention in the literature and made its way into popular books [9, 42], until it was answered affirmatively by Mütze [23]; see also [13].

Similarly to permutations, there are also many known methods for generating general (α,β)-combinations that use operations other than star transpositions, see [5, 11, 17, 29, 38]. In particular, (α,β)-combinations can be generated by adjacent transpositions if and only if $\alpha=1$, $\beta=1$, or α and β are both odd [3, 10, 26].

1.3 Multiset permutations

Shen and Williams [34] proposed a far-ranging generalization of the middle levels conjecture that connects permutations and combinations. Their conjecture is about multiset permutations. For integers $k \geq 2$ and $a_1,\ldots,a_k \geq 1$, an $(a_1,\ldots,a_k)$-*multiset permutation* is a string over the alphabet $\{1,\ldots,k\}$ that contains exactly a_i occurrences of the symbol i. We refer to the sequence $\boldsymbol{a} := (a_1,\ldots,a_k)$ as the *frequency vector*, as it specifies the frequency of each symbol. The length of a multiset permutation is $n_{\boldsymbol{a}} := a_1+\cdots+a_k$, and if the context is clear we omit the index and simply write $n = n_{\boldsymbol{a}}$. If all symbols appear equally often, i.e., $a_1=\cdots=a_k=\alpha$, we use the abbreviation $\alpha^k := (a_1,\ldots,a_k)$. For example 123433153 is a $(2,1,4,1)$-multiset permutation, and 331232142144 is a 3^4-multiset permutation.

Clearly, multiset permutations are a generalization of permutations and combinations. Specifically, k-permutations are 1^k-multiset permutations, and (α,β)-combinations are (α,β)-multiset permutations (up to shifting the symbol names $1,2 \mapsto 0,1$). Stachowiak [37] showed that $(a_1,\ldots,a_k)$-multiset permutations can be generated by adjacent transpositions if and only if at least two of the a_i are odd.



(a) $(a_1, a_2) = (4, 4)$

(b) $(a_1, a_2, a_3) = (2, 2, 2)$

Figure 2 Star transposition Gray codes for (a) $(4,4)$- and (b) $(2,2,2)$-multiset permutations. The strings are arranged in clockwise order, starting at 12 o'clock, with the first entry on the inner track, and the last entry on the outer track. As every star transposition changes the first entry, the color on the inner track changes in every step.

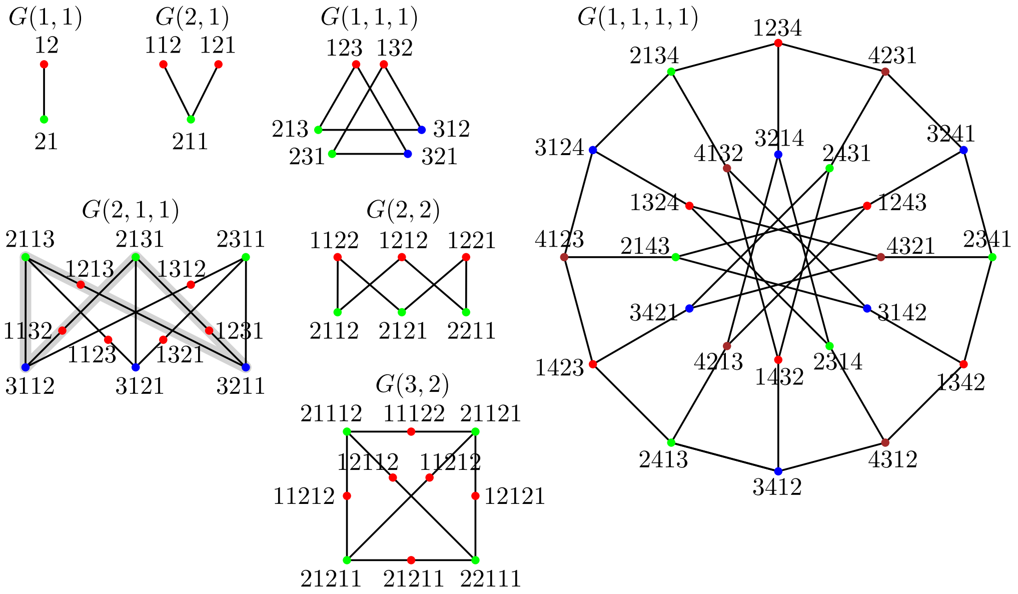


Figure 3 Star transposition graphs $G(\boldsymbol{a})$ for several small multiset permutations $\boldsymbol{a}$. Vertices are colored according to the first entry of the multiset permutations, and these color classes form independent sets. In $G(2,1,1)$, an odd cycle is highlighted.

Shen and Williams [34] conjectured that all α^k-multiset permutations can be generated by star transpositions, for any $\alpha \geq 1$ and $k \geq 2$. We state their conjecture in terms of Hamilton cycles in a suitably defined graph, as follows. We write $\Pi(\boldsymbol{a}) = \Pi(a_1, \ldots, a_k)$ for the set of all $(a_1, \ldots, a_k)$-multiset permutations. Moreover, we let $G(\boldsymbol{a}) = G(a_1, \ldots, a_k)$ denote the graph on the vertex set $\Pi(\boldsymbol{a}) = \Pi(a_1, \ldots, a_k)$ with an edge between any two multiset permutations that differ in a star transposition, i.e., in swapping the first entry of the multiset permutation with any entry at positions $2, \ldots, n$ that is distinct from the first one. Figure 3 shows various examples of the graph $G(\boldsymbol{a})$. When denoting specific multiset permutations we sometimes omit commas and brackets for brevity, for example $1312214 \in \Pi(3, 2, 1, 1)$.

▶ **Conjecture 1** ([34]). *For any $\alpha \geq 1$ and $k \geq 2$, the graph $G(\alpha^k)$ has a Hamilton cycle.*

In this and the following statements, the single edge $G(1, 1)$ is also considered a cycle, as it gives a cyclic Gray code. Note that $G(a_1, \ldots, a_k)$ is vertex-transitive if and only if $a_1 = \cdots = a_k =: \alpha$. In this case, Conjecture 1 is an interesting instance of the aforementioned conjecture of Lovász [21] on Hamilton paths in vertex-transitive graphs.

Evidence for Conjecture 1 comes from the results mentioned in Sections 1.1 and 1.2 on generating permutations by star transpositions and the solution of the middle levels conjecture, respectively, formulated in terms of the graph $G(\boldsymbol{a})$ below. These known results settle the boundary cases $\alpha = 1$ and $k \geq 2$, and $\alpha \geq 1$ and $k = 2$, respectively, of Conjecture 1.

▶ **Theorem 2** (Ehrlich; [20]; [36]). *For any $k \geq 2$, the graph $G(1^k)$ has a Hamilton cycle.*

▶ **Theorem 3** ([23, 13]). *For any $\alpha \geq 1$, the graph $G(\alpha, \alpha)$ has a Hamilton cycle.*

In their paper, Shen and Williams also provided an ad-hoc solution for the first case of their conjecture that is not covered by Theorems 2 and 3, namely a Hamilton cycle in $G(2, 2, 2)$, which is displayed in Figure 2 (b).

We approach Conjecture 1 by tackling the following even more general question: For which frequency vectors $\boldsymbol{a} = (a_1, \ldots, a_k)$ does the graph $G(\boldsymbol{a})$ have a Hamilton cycle? By renaming symbols, we may assume w.l.o.g. that the entries of the vector $\boldsymbol{a}$ are non-increasing, i.e.,

$$a_1 \geq a_2 \geq \cdots \geq a_k. \tag{1}$$

We can thus think of the vector $\boldsymbol{a}$ as an integer partition of n.

2 Our results

For any $i \in [n]$ and $c \in [k]$, we write $\Pi(\boldsymbol{a})^{i,c}$ for the set of all multiset permutations from $\Pi(\boldsymbol{a})$ whose ith symbol equals c. Note that every star transposition changes the first entry; see the inner track of each of the two wheels in Figure 2. As a consequence, $G(\boldsymbol{a})$ is a k-partite graph with partition classes $\Pi(\boldsymbol{a})^{1,1}, \ldots, \Pi(\boldsymbol{a})^{1,k}$; see Figure 3. Moreover, the partition class $\Pi(\boldsymbol{a})^{1,1}$ is a largest one because of (1). This k-partition of the graph $G(\boldsymbol{a})$ is a potential obstacle for the existence of Hamilton cycles and paths. Specifically, if one partition class is larger than all others combined, then there cannot be a Hamilton cycle, and if the size difference is more than 1, then there cannot be a Hamilton path.

We capture this by defining a parameter $\Delta(\boldsymbol{a})$ for any integer partition $\boldsymbol{a} = (a_1, \ldots, a_k)$ as

$$\Delta(\boldsymbol{a}) := n - 2a_1 = -a_1 + \sum_{i=2}^{k} a_i. \tag{2}$$

We will see that if $\Delta(\boldsymbol{a}) < 0$, then the partition class $\Pi(\boldsymbol{a})^{1,1}$ of the graph $G(\boldsymbol{a})$ is larger than all others combined, excluding the existence of Hamilton cycles. On the other hand, if $\Delta(\boldsymbol{a}) \geq 0$, then every partition class of the graph $G(\boldsymbol{a})$ is at most as large as all others combined (equality holds if $\Delta(\boldsymbol{a}) = 0$), which does not exclude the existence of a Hamilton cycle. The cases with $\Delta(\boldsymbol{a}) = 0$ lie on the boundary between the two regimes, and they are the hardest in terms of proving Hamiltonicity. These cases can be seen as generalizations of the middle levels conjecture, namely the case $\boldsymbol{a} = (\alpha, \alpha)$ captured by Theorem 3, which also satisfies $\Delta(\boldsymbol{a}) = 0$.

▶ **Theorem 4.** *For any integer partition $\boldsymbol{a} = (a_1, \ldots, a_k)$ with $\Delta(\boldsymbol{a}) < 0$ the graph $G(\boldsymbol{a})$ does not have a Hamilton cycle, and it does not have a Hamilton path unless $\boldsymbol{a} = (2, 1)$.*

For $k = 2$ symbols, the condition $\Delta(\boldsymbol{a}) < 0$ is equivalent to $a_1 > a_2$, i.e., there is no star transposition Gray code for "unbalanced" combinations.

We now discuss the cases $\Delta(\boldsymbol{a}) \geq 0$. Our first main goal is to reduce all cases with $\Delta(\boldsymbol{a}) > 0$ to cases with $\Delta(\boldsymbol{a}) = 0$. For doing so, it is helpful to consider stronger notions of Hamiltonicity. Specifically, we consider Hamilton-connectedness and Hamilton-laceability, which have been heavily studied (see [1, 2, 6, 14, 16, 35]). A graph is called *Hamilton-connected* if there is a Hamilton path between any two distinct vertices. A bipartite graph is called *Hamilton-laceable* if there is a Hamilton path between any pair of vertices from the two partition classes. In general, the graphs $G(\boldsymbol{a})$ are not bipartite, so we say that $G(\boldsymbol{a})$ is *1-laceable* if there is a Hamilton path between any vertex in $\Pi(\boldsymbol{a})^{1,1}$ and any vertex not in $\Pi(\boldsymbol{a})^{1,1}$, i.e., between any vertex with first symbol 1 and any vertex with first symbol distinct from 1.

This approach is inspired by the following result of Tchuente [39], who strengthened Theorem 2 considerably.

▶ **Theorem 5** ([39])**.** *For any $k \geq 4$, the graph $G(1^k)$ is Hamilton-laceable.*

The key insight is that proving a stronger property makes the proof easier and shorter, because the inductive statement is more powerful and flexible; see Section 3.2 below. Encouraged by this, we raise the following conjecture about graphs $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) = 0$. It is another natural and far-ranging generalization of the middle levels conjecture, which we support by extensive computer experiments and by proving some special cases.

▶ **Conjecture 6.** *For any integer partition $\boldsymbol{a} = (a_1, \ldots, a_k)$ with $\Delta(\boldsymbol{a}) = 0$ the graph $G(\boldsymbol{a})$ is Hamilton-1-laceable, unless $\boldsymbol{a} = (2, 2)$.*

The exceptional graph $G(2, 2)$ mentioned in this conjecture is a 6-cycle; see Figure 3. Assuming the validity of this conjecture, we settle all cases $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) > 0$ in the strongest possible sense. While being a conditional result, the main purpose of this theorem is to reduce all cases $\Delta(\boldsymbol{a}) \geq 0$ to the boundary cases $\Delta(\boldsymbol{a}) = 0$.

▶ **Theorem 7.** *Conditional on Conjecture 6, for any integer partition $\boldsymbol{a} = (a_1, \ldots, a_k)$ with $\Delta(\boldsymbol{a}) > 0$ the graph $G(\boldsymbol{a})$ is Hamilton-connected, unless $\boldsymbol{a} = (1, 1, 1)$ or $\boldsymbol{a} = 1^k$ for $k \geq 4$, and possibly unless $\boldsymbol{a} = (\alpha, \alpha, 1)$ for $\alpha \geq 3$.*

The dependence of Theorem 7 on Conjecture 6 can be captured more precisely. Specifically, $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) > 0$ is shown to be Hamilton-connected, assuming that $G(\boldsymbol{b})$ with $\Delta(\boldsymbol{b}) = 0$ is Hamilton-1-laceable for all integer partitions $\boldsymbol{b}$ that are majorized componentwise by $\boldsymbol{a}$.

The three exceptions mentioned in Theorem 7 are well understood: Specifically, $G(1, 1, 1)$ is a 6-cycle; see Figure 3. Furthermore, $G(1^k)$ for $k \geq 4$ is Hamilton-laceable by Theorem 5. Lastly, we will show that $G(\alpha, \alpha, 1)$ for $\alpha \geq 3$ satisfies a variant of Hamilton-laceability, which also guarantees a Hamilton cycle. In fact, we believe that $G(\alpha, \alpha, 1)$ is Hamilton-connected, but we cannot prove it.

We provide the following evidence for Conjecture 6. First of all, with computer help we verified that $G(\boldsymbol{a})$ is indeed Hamilton-1-laceable for all integer partitions $\boldsymbol{a} \neq (2,2)$ with $\Delta(\boldsymbol{a}) = 0$ that satisfy $n \leq 8$, i.e., for $\boldsymbol{a} \in \{(1,1),(2,1,1),(3,3),(3,2,1),(3,1,1,1),(4,4),(4,3,1),(4,2,2),(4,2,1,1),(4,1,1,1,1)\}$. Furthermore, we prove the case of $k = 2$ symbols unconditionally. Note that for $k = 2$, Hamilton-1-laceability is the same as Hamilton-laceability. Recall that $G(\alpha,\alpha)$ is isomorphic to the subgraph of the $(2\alpha-1)$-dimensional hypercube induced by the middle two levels, so the following result is a considerable strengthening of Theorem 3, the middle levels theorem.

▶ **Theorem 8.** *For any $\alpha \geq 3$, the graph $G(\alpha,\alpha)$ is Hamilton-laceable.*

We also have the following (unconditional) result for $k = 3$ symbols.

▶ **Theorem 9.** *For any $\alpha \geq 2$, the graph $G(\alpha,\alpha-1,1)$ has a Hamilton cycle.*

Lastly, we consider integer partitions $\boldsymbol{a} = (a_1,\ldots,a_k)$, $k \geq 3$, with $\Delta(\boldsymbol{a}) \geq 0$ and an upper bound on the part size, i.e., $a_1 \leq \alpha$ for some constant α. By the remarks after Theorem 7, the inductive proof of the theorem for such integer partitions only relies on Conjecture 6 being satisfied for integer partitions with the same upper bound on the part size. For any fixed bound α, there are only finitely many such partitions with $\Delta(\boldsymbol{a}) = 0$ that can be checked by computer. For example, for $\alpha = 4$ these are $\boldsymbol{a} \in \{(2,1,1),(3,2,1),(3,1,1,1),(4,3,1),(4,2,2),(4,2,1,1),(4,1,1,1,1)\}$. This yields the following (unconditional) result.

▶ **Theorem 10.** *For $\alpha \in \{2,3,4\}$ and any integer partition $\boldsymbol{a} = (a_1,\ldots,a_k)$ with $\Delta(\boldsymbol{a}) > 0$ and $a_1 = \alpha$, the graph $G(\boldsymbol{a})$ is Hamilton-connected.*

In words, Theorem 10 settles all integer partitions $\boldsymbol{a}$ whose largest part is at most 4. In particular, this settles the cases $\alpha \in \{2,3,4\}$ and $k \geq 2$ of Shen and Williams' Conjecture 1 in a rather strong sense.

3 Proof ideas

In this section, we give a high-level overview of the main ideas and techniques used in our proofs. No formal proofs of our results are presented in this extended abstract due to space constraints, but they can be found in the preprint [12].

3.1 The case $\Delta(\boldsymbol{a}) < 0$

The main idea for proving Theorem 4 is that if $\Delta(\boldsymbol{a}) < 0$, then the partition class $\Pi(\boldsymbol{a})^{1,1}$ of the graph $G(\boldsymbol{a})$ is larger than all others combined, which excludes the existence of a Hamilton cycle. To exclude the existence of a Hamilton path, we show that the size difference is strictly more than 1, unless $\boldsymbol{a} = (2,1)$. Note that the graph $G(2,1)$ is the path on three vertices, so in this case the size difference is precisely 1. These arguments are based on straightforward algebraic manipulations involving multinomial coefficients.

3.2 The case $\Delta(\boldsymbol{a}) > 0$

To prove Theorem 7, it is convenient to think of an integer partition $\boldsymbol{a} = (a_1,\ldots,a_k)$ as an infinite non-increasing sequence $(a_1,a_2,\ldots)$, with only k nonzero entries at the beginning. Given two such integer partitions $\boldsymbol{a} = (a_1,a_2,\ldots)$ and $\boldsymbol{b} = (b_1,b_2,\ldots)$, we write $\boldsymbol{b} \prec \boldsymbol{a}$ if $b_i \leq a_i$ for all $i \geq 1$. Integer partitions with the partial order $\prec$ form a lattice, which is the

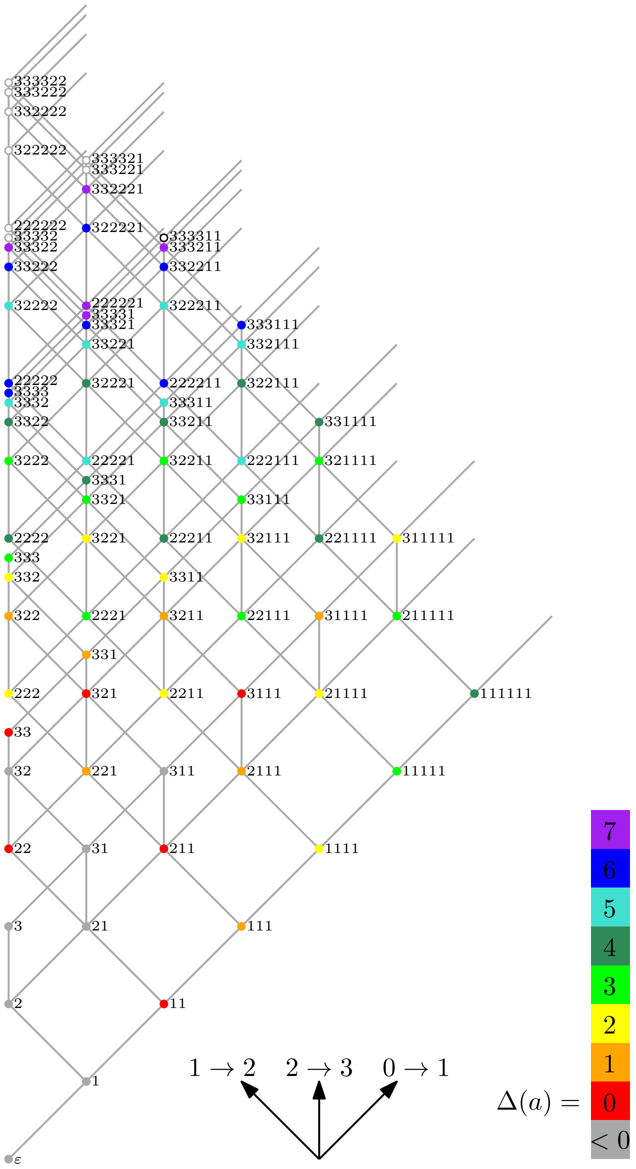


Figure 4 The lattice of integer partitions $\boldsymbol{a} = (a_1, \ldots, a_k)$ with largest part $a_1 \leq 3$. The coordinates are projected into three dimensions depending on which value is increased.

sublattice of the infinite lattice $\mathbb{N}^{\mathbb{N}}$ cut out by the hyperplanes defined by (1); see Figure 4. The cover relations in this lattice are given by decrementing any of the a_i for which $a_i > a_{i+1}$. We write $\boldsymbol{b} \prec\!\!\cdot\, \boldsymbol{a}$ for partitions $\boldsymbol{a} \neq \boldsymbol{b}$ if $\boldsymbol{b} \prec \boldsymbol{a}$ and there is no $\boldsymbol{c} \notin \{\boldsymbol{a}, \boldsymbol{b}\}$ with $\boldsymbol{b} \prec \boldsymbol{c} \prec \boldsymbol{a}$.

In this lattice of integer partitions, the hyperplane defined by $\Delta(\boldsymbol{a}) = 0$ separates the cases where Hamiltonicity is impossible, which lie on the side of the hyperplane where $\Delta(\boldsymbol{a}) < 0$ (Theorem 4), from the cases where Hamiltonicity can be established more easily, which lie on the side of the hyperplane where $\Delta(\boldsymbol{a}) > 0$ (Theorem 7). The cases $\Delta(\boldsymbol{a}) = 0$ on the hyperplane are the hardest ones (Conjecture 6).

Our proof of Theorem 7 proceeds by induction in this partition lattice and establishes the Hamiltonicity of $G(\boldsymbol{a})$ by using the Hamiltonicity of $G(\boldsymbol{b})$ for all integer partitions $\boldsymbol{b} \prec\!\!\cdot\, \boldsymbol{a}$, where Conjecture 6 serves as the base case of the induction. This is based on the observation that fixing one of the symbols at positions $2, \ldots, n$ in $G(\boldsymbol{a})$ yields subgraphs that are isomorphic to $G(\boldsymbol{b})$ for $\boldsymbol{b} \prec\!\!\cdot\, \boldsymbol{a}$.

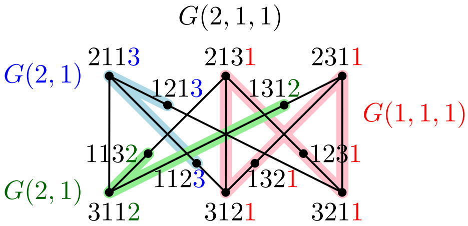


Figure 5 Decomposing $G(2,1,1)$ into three subgraphs by fixing the last symbol.

Specifically, for any $\boldsymbol{a} = (a_1, \ldots, a_k)$, $i = 2, \ldots, n$, and $c \in [k]$, the subgraph of $G(\boldsymbol{a})$ induced by the vertex set $\Pi(\boldsymbol{a})^{i,c}$ is isomorphic to $G(\boldsymbol{b})$ where $\boldsymbol{b}$ is the partition obtained from $\boldsymbol{a}$ by decreasing a_i by 1 (and possibly sorting the resulting numbers non-increasingly); see Figure 5.

Moreover, for any $\boldsymbol{b} \prec \boldsymbol{a}$ we have $\Delta(\boldsymbol{b}) = \Delta(\boldsymbol{a}) - 1$ or $\Delta(\boldsymbol{b}) = \Delta(\boldsymbol{a}) + 1$. In particular, if $\Delta(\boldsymbol{a}) > 0$, then we have $\Delta(\boldsymbol{b}) \geq 0$. For example, the vertex set of $G(\boldsymbol{a})$ for $\boldsymbol{a} = (3,2,2)$ ($\Delta(\boldsymbol{a}) = 1$) can be partitioned into one copy of $G(\boldsymbol{b})$ for $\boldsymbol{b} = (2,2,2)$ (if the fixed symbol is $c = 1$; $\Delta(\boldsymbol{b}) = 2$) and two copies of $G(\boldsymbol{b}')$ for $\boldsymbol{b}' = (3,2,1)$ (if the fixed symbol is $c = 2$ or $c = 3$; $\Delta(\boldsymbol{b}') = 0$). Therefore, we may construct a Hamilton path in $G(3,2,2)$ by gluing together paths in each of these three subgraphs which exist by induction.

While conceptually simple, implementing this idea incurs considerable technical obstacles, in particular for some of the graphs $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) = 1$, i.e., instances that are very close to the hyperplane $\Delta(\boldsymbol{a}) = 0$. The proof is split into several interdependent lemmas, and it is the technically most demanding part of our work.

Theorem 10 follows immediately from the inductive proof of Theorem 7 and by settling finitely many cases with computer help.

To further illustrate the ideas outlined before, we close this section by reproducing Tchuente's proof of Theorem 5.

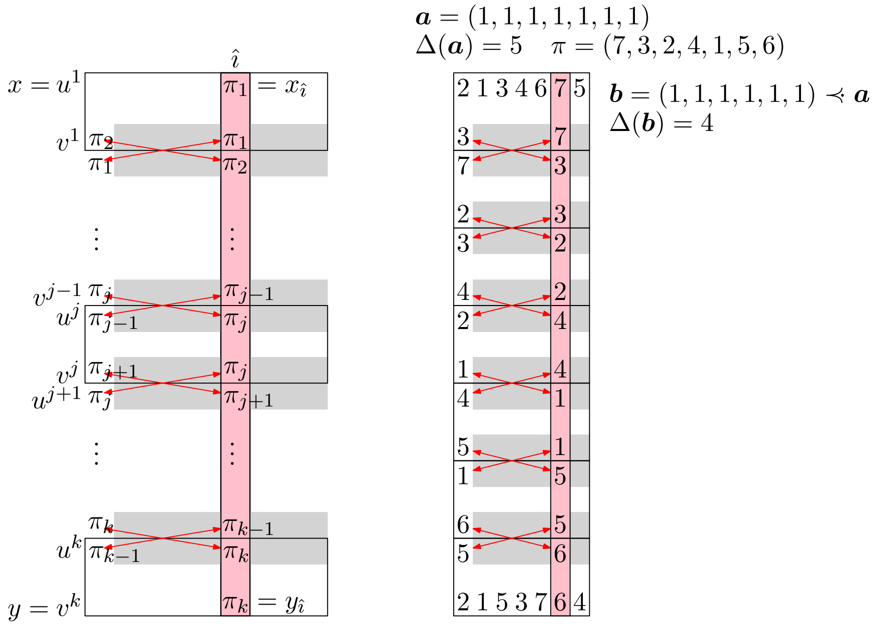


Figure 6 Illustration of the proof of Theorem 5. The left hand side shows the general schematic partitioning of the graph $G(1^k)$ into blocks, each of which is a copy of $G(1^{k-1})$, by fixing symbols at position $\hat{\imath}$. The right hand side shows a concrete example for $k = 7$.

Proof of Theorem 5. To prove that $G(1^k)$ is Hamilton-laceable, we proceed by induction on k. The induction basis $k = 4$ can be checked by straightforward case analysis. For the induction step, we assume that $G(1^{k-1})$, $k \geq 5$, is Hamilton-laceable, and we prove that $G(1^k)$ is also Hamilton-laceable. Note that $1^{k-1} \prec\!\!\cdot\; 1^k$ and $\Delta(1^k) = k - 2 = \Delta(1^{k-1}) + 1$. The following arguments are illustrated in Figure 6. The partition classes of the graph $G(1^k)$ are given by the parity of the permutations, i.e., by the number of inversions. Therefore, we consider two distinct permutations x and y of $[k]$ with opposite parity, and we need to show how to connect them by a Hamilton path in $G(1^k)$. As $x \neq y$, there is a position $\hat{\imath} > 1$ in which x and y differ, i.e., $x_{\hat{\imath}} \neq y_{\hat{\imath}}$, and this is the position that we will fix to different symbols. Specifically, we choose a permutation π of $[k]$ such that $\pi_1 = x_{\hat{\imath}}$ and $\pi_k = y_{\hat{\imath}}$. The permutation π captures the order in which we will fix symbols at position $\hat{\imath}$. We then choose permutations u^j, v^j of $[k]$, $j = 1, \ldots, k$, satisfying $u^1 = x$, $v^k = y$, and such that u^j is obtained from v^{j-1} by a star transposition of the symbol π_j at position 1 with the symbol π_{j-1} at position $\hat{\imath}$, for all $j = 2, \ldots, k$. Moreover, we choose u^j and v^j such that the parity of the number of inversions after removing the symbol π_j is opposite. Specifically, for u^j this parity is the same as for $u^1 = x$ if and only if π_j has the same parity as π_1, and for v^j this parity is the same as for $v^k = y$ if and only if π_j has the same parity as π_k. For each $j = 1, \ldots, k$, we now consider the permutations whose $\hat{\imath}$th entry equals π_j (formally, this is the set $\Pi(1^k)^{\hat{\imath}, \pi_j}$). Clearly, the subgraph of $G(1^k)$ induced by these permutations is isomorphic to $G(1^{k-1})$. In other words, by removing the $\hat{\imath}$th entry and renaming entries to $1, \ldots, k - 1$, we obtain permutations of $[k - 1]$. Consequently, by induction there is a path in $G(1^k)$ that visits all permutations whose $\hat{\imath}$th entry equals π_j and that connects u^j to v^j. The concatenation of those k paths obtained by induction is the desired Hamilton path in $G(1^k)$ from x to y, which completes the induction step. ◀

Note that the constraints imposed on the permutations π and u^j, v^j, $j = 1, \ldots, k$, in this proof are very mild, and leave a lot of room for modifications to construct many different Hamilton paths, possibly so as to satisfy some additional conditions.

3.3 The case $\Delta(\boldsymbol{a}) = 0$

Our proofs of Theorems 8 and 9 build on ideas introduced in the papers [13, 22].

Specifically, the first step in proving Theorem 8 is to build a cycle factor in the graph $G(\alpha, \alpha)$, i.e., a collection of disjoint cycles in the graph that together visit all vertices. We then choose vertices x and y from the two partition classes of the graph that we want to connect by a Hamilton path. In this we can take into account automorphisms of $G(\alpha, \alpha)$, i.e., for proving laceability only certain pairs of vertices x and y in the two partition classes have to be considered. In the next step, we join a small subset of cycles from the factor, including the ones containing x and y, to a short path between x and y. This is achieved by taking the symmetric difference of the edge set of the cycle factor with a carefully chosen path P from x to y that alternately uses edges on one of the cycles from the factor and edges that go between two such cycles; see Figure 7 (a)+(b). In the last step, we join the remaining cycles of the factor to the path between x and y, until we end with a Hamilton path from x to y. Each such joining is achieved by taking the symmetric difference of the cycle factor with a suitably chosen 6-cycle; see Figure 7 (b)+(c).

It was shown in [13] that the cycles of the aforementioned cycle factor in $G(\alpha, \alpha)$ are bijectively equivalent to plane trees with α vertices, and the joining operations via 6-cycles can be interpreted combinatorially as local change operations between two such plane trees. To prove Theorem 9, we first generalize the construction of this cycle factor in the graph $G(\alpha, \alpha)$ to a cycle factor in any graph $G(\boldsymbol{a})$, $\boldsymbol{a} = (a_1, \ldots, a_k)$, with $\Delta(\boldsymbol{a}) = 0$. It turns out that the cycles of this generalized factor can be interpreted combinatorially as *vertex-labeled*

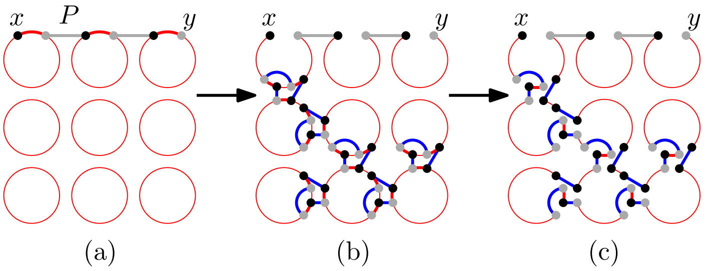


Figure 7 Strategy of the proof of Theorem 8.

plane trees, where exactly a_i vertices have the label i for $i = 2, \ldots, k$. If $\boldsymbol{a} = (\alpha, \alpha)$, then all vertex labels are the same, so we can consider the trees as unlabeled. Moreover, the joining 6-cycles in $G(\alpha, \alpha)$ generalize nicely to joining 12- or 6-cycles in $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) = 0$, and they correspond to local change operations involving sets T of labeled plane trees with $|T| \in \{2, 3, 5, 6\}$, depending on the location of vertex labels. For proving Theorem 9, we consider the special case $\boldsymbol{a} = (\alpha, \alpha - 1, 1)$, i.e., exactly one vertex in the plane trees is labeled differently from all other vertices, and we show that there is a choice of joining cycles so that the symmetric difference with the cycle factor yields a Hamilton cycle in the graph $G(\boldsymbol{a})$.

4 Open questions

We conclude this paper with the following open questions.

- We believe that the ideas outlined in Section 3.3 to prove that $G(\alpha, \alpha - 1, 1)$ has a Hamilton cycle are in principle suitable to prove that $G(\boldsymbol{a})$ has a Hamilton cycle for all $\boldsymbol{a}$ with $\Delta(\boldsymbol{a}) = 0$, which would be an important first step towards a proof of Conjecture 6. In particular, the construction of the cycle factor and gluing tuples based on vertex-labeled plane trees are fully general. The main difficulty in combining these ingredients lies in the fact that some gluing cycles join more than two cycles from the factor (namely 3, 5, or 6 cycles) to a single cycle, and in this case the resulting interactions between different gluing cycles seem to be hard to control.
- We conjecture that $G(\alpha, \alpha, 1)$ for $\alpha \geq 2$ is Hamilton-connected, just as all other graphs $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) > 0$ covered by Theorem 7. However, we are unable to prove this based on Conjecture 6. With computer help, we verified that $G(\alpha, \alpha, 1)$ is Hamilton-connected for $\alpha = 2, 3, 4$. Proving this in general would streamline our proof of Theorem 7 considerably, as it would make several auxiliary lemmas redundant. To prove that $G(\alpha, \alpha, 1)$ is Hamilton-connected, it may help to establish a Hamiltonicity property for graphs $G(\boldsymbol{a})$ with $\Delta(\boldsymbol{a}) = 0$ and $k \geq 3$ that is stronger than 1-laceability. Specifically, in addition to a Hamilton path between any vertex in $\Pi(\boldsymbol{a})^{1,1}$ and any vertex not in $\Pi(\boldsymbol{a})^{1,1}$, we may also ask for a Hamilton path between any two distinct vertices in $\Pi(\boldsymbol{a})^{1,1}$. We checked by computer whether $G(\boldsymbol{a})$ has this stronger property for $\boldsymbol{a} \in \{(2,1,1), (3,2,1), (3,1,1,1), (4,3,1), (4,2,2), (4,2,1,1), (4,1,1,1,1)\}$, and it was satisfied in all cases except for $\boldsymbol{a} = (2,1,1)$.
- While our proofs are constructive, they are far from yielding efficient algorithms for computing the corresponding Gray codes. Ideally, one would like algorithms whose running time is polynomial in n per generated multiset permutation of length n. Such algorithms are known for the Hamilton cycles mentioned in Theorems 2 and 3, see [19, Section 7.2.1.2] and [24], respectively. An interesting direction could be to explore greedy algorithms for generating multiset permutations by star transpositions, which may yield much simpler constructions to start with, cf. [41, 4, 31].

- Knuth raised the question whether there are star transposition Gray codes for (α, α)-combinations whose flip sequence can be partitioned into $2\alpha - 1$ blocks, such that each block is obtained from the previous one by adding $+1$ modulo $2\alpha - 1$. This problem is a strengthening of the middle levels conjecture, and it was answered affirmatively in [22]. The Gray code for $(4, 4)$-combinations shown in Figure 2 (a) has such a 7-fold cyclic symmetry. We can ask more generally: Are there star transposition Gray codes for multiset permutations whose flip sequence can be partitioned into $n - 1$ blocks, such that each block is obtained from the previous one by adding $+1$ modulo $n - 1$? Figure 8 shows an ad-hoc solution for $(2, 2, 2)$-multiset permutations with 5-fold cyclic symmetry.
- A more general version of the problem considered in this paper is the following: We consider an alphabet $\{1, \dots, k\}$ of size $k \geq 2$, and frequencies $a_1, \dots, a_k \geq 1$ that specify that symbol i appears exactly a_i times for all $i = 1, \dots, k$. Moreover, there is an additional integer parameter s with $1 \leq s \leq k - 1$ that has the following significance. The objects to be generated are all pairs (S, x), where S is a set or string of s distinct symbols, and x is a string of the remaining $n - s$ symbols, where $n := a_1 + \cdots + a_k$. A star transposition swaps one symbol from S with one symbol from x that is currently not in S, and the question is whether there is a star transposition Gray code for all those objects.

  Note that multiset permutations considered in this paper are the special case when $s = 1$. Andrea Sportiello suggested this problem with $a_1 = \cdots = a_k = \alpha$ (uniform frequency) and S being a set as a generalization of the middle levels conjecture ($s = 1$, $k = 2$, $a_1 = a_2 = \alpha$). Moreover, Ajit A. Diwan suggested this problem with $a_1 = \cdots = a_k = \alpha$ (uniform frequency) and a set S of size $s = k - 1$. Note that the uniform frequency case is particularly interesting, as the underlying flip graph for this problem is vertex-transitive if and only if $a_1 = \cdots = a_k$ (recall Lovász' conjecture [21]).

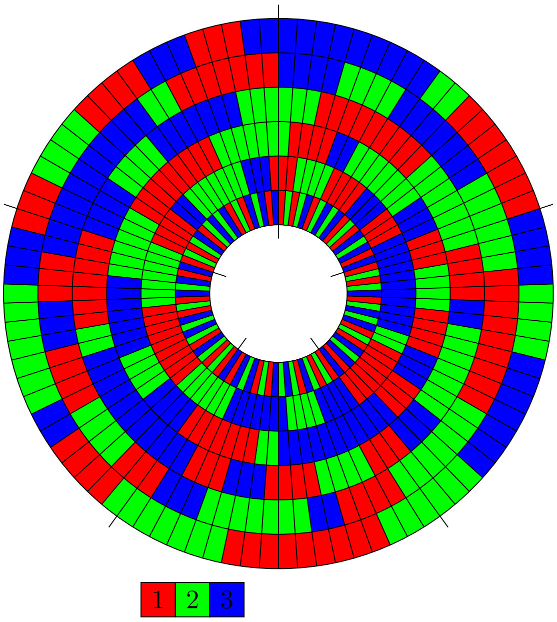


Figure 8 Star transposition Gray code for $(2, 2, 2)$-multiset permutations with 5-fold cyclic symmetry.

References

1 B. Alspach and Y. Qin. Hamilton-connected Cayley graphs on Hamiltonian groups. *European J. Combin.*, 22(6):777–787, 2001. doi:10.1006/eujc.2001.0456.

2 T. Araki. Hyper Hamiltonian laceability of Cayley graphs generated by transpositions. *Networks*, 48(3):121–124, 2006. doi:10.1002/net.20126.

3 M. Buck and D. Wiedemann. Gray codes with restricted density. *Discrete Math.*, 48(2-3):163–171, 1984. doi:10.1016/0012-365X(84)90179-1.

4 B. Cameron, J. Sawada, and A. Williams. A Hamilton cycle in the k-sided pancake network, 2021. arXiv:2103.09256.

5 P. Chase. Combination generation and graylex ordering. *Congr. Numer.*, 69:215–242, 1989. Eighteenth Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1988).

6 C. C. Chen and N. F. Quimpo. On strongly Hamiltonian abelian group graphs. In *Combinatorial mathematics, VIII (Geelong, 1980)*, volume 884 of *Lecture Notes in Math.*, pages 23–34. Springer, Berlin-New York, 1981.

7 R. C. Compton and S. G. Williamson. Doubly adjacent Gray codes for the symmetric group. *Linear and Multilinear Algebra*, 35(3-4):237–293, 1993. doi:10.1080/03081089308818261.

8 P. F. Corbett. Rotator graphs: An efficient topology for point-to-point multiprocessor networks. *IEEE Transactions on Parallel and Distributed Systems*, 3:622–626, 1992.

9 P. Diaconis and R. Graham. *Magical mathematics.* Princeton University Press, Princeton, NJ, 2012. The mathematical ideas that animate great magic tricks, With a foreword by Martin Gardner.

10 P. Eades, M. Hickey, and R. C. Read. Some Hamilton paths and a minimal change algorithm. *J. Assoc. Comput. Mach.*, 31(1):19–29, 1984. doi:10.1145/2422.322413.

11 P. Eades and B. McKay. An algorithm for generating subsets of fixed size with a strong minimal change property. *Inform. Process. Lett.*, 19(3):131–133, 1984. doi:10.1016/0020-0190(84)90091-7.

12 P. Gregor, A. Merino, and T. Mütze. arXiv:2108.07465.

13 P. Gregor, T. Mütze, and J. Nummenpalo. A short proof of the middle levels theorem. *Discrete Analysis*, 2018:8:12 pp., 2018.

14 F. Harary and M. Lewinter. Hypercubes and other recursively defined Hamilton laceable graphs. *Congr. Numer.*, 60:81–84, 1987. Eighteenth Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, Fla., 1987).

15 I. Havel. Semipaths in directed cubes. In *Graphs and other combinatorial topics (Prague, 1982)*, volume 59 of *Teubner-Texte Math.*, pages 101–108. Teubner, Leipzig, 1983.

16 S.-Y. Hsieh, G.-H. Chen, and C.-W. Ho. Hamiltonian-laceability of star graphs. *Networks*, 36(4):225–232, 2000. doi:10.1002/1097-0037(200012)36:4<225::AID-NET3>3.0.CO;2-G.

17 T. Jenkyns and D. McCarthy. Generating all k-subsets of $\{1\cdots n\}$ with minimal changes. *Ars Combin.*, 40:153–159, 1995.

18 S. Johnson. Generation of permutations by adjacent transposition. *Math. Comp.*, 17:282–285, 1963.

19 D. E. Knuth. *The Art of Computer Programming. Vol. 4A. Combinatorial Algorithms. Part 1.* Addison-Wesley, Upper Saddle River, NJ, 2011.

20 V. L. Kompel'maher and V. A. Liskovec. Sequential generation of permutations by means of a transposition basis. *Kibernetika (Kiev)*, 3:17–21, 1975.

21 L. Lovász. Problem 11. In *Combinatorial Structures and Their Applications (Proc. Calgary Internat. Conf., Calgary, Alberta, 1969)*. Gordon and Breach, New York, 1970.

22 A. I. Merino, O. Mička, and T. Mütze. On a combinatorial generation problem of Knuth. In D. Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 735–743. SIAM, 2021. doi:10.1137/1.9781611976465.46.

23 T. Mütze. Proof of the middle levels conjecture. *Proc. Lond. Math. Soc.*, 112(4):677–713, 2016. doi:10.1112/plms/pdw004.

24 T. Mütze and J. Nummenpalo. A constant-time algorithm for middle levels Gray codes. *Algorithmica*, 82(5):1239–1258, 2020. doi:10.1007/s00453-019-00640-2.

25 A. Nijenhuis and H. Wilf. *Combinatorial algorithms.* Academic Press, New York-London, 1975. Computer Science and Applied Mathematics.

26 F. Ruskey. Adjacent interchange generation of combinations. *J. Algorithms*, 9(2):162–180, 1988. doi:10.1016/0196-6774(88)90036-3.

27 F. Ruskey. Combinatorial Gray code. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*, pages 342–347. Springer, 2016.

28 F. Ruskey and C. D. Savage. Hamilton cycles that extend transposition matchings in Cayley graphs of S_n. *SIAM J. Discrete Math.*, 6(1):152–166, 1993. doi:10.1137/0406012.

29 F. Ruskey and A. Williams. The coolest way to generate combinations. *Discrete Math.*, 309(17):5305–5320, 2009. doi:10.1016/j.disc.2007.11.048.

30 C. D. Savage. A survey of combinatorial Gray codes. *SIAM Rev.*, 39(4):605–629, 1997. doi:10.1137/S0036144595295272.

31 J. Sawada and A. Williams. Greedy flipping of pancakes and burnt pancakes. *Discrete Appl. Math.*, 210:61–74, 2016. doi:10.1016/j.dam.2016.02.005.

32 J. Sawada and A. Williams. Solving the sigma-tau problem. *ACM Trans. Algorithms*, 16(1):Art. 11, 17 pp., 2020. doi:10.1145/3359589.

33 X. S. Shen and A. Williams. A 'hot potato' Gray code for permutations. *Electronic Notes in Discrete Mathematics*, 44:89–94, 2013. doi:10.1016/j.endm.2013.10.014.

34 X. S. Shen and A. Williams. A k-ary middle levels conjecture. In *Proceedings of the 23rd Thailand-Japan Conference on Discrete and Computational Geometry, Graphs, and Games*, 2021.

35 G. J. Simmons. Almost all n-dimensional rectangular lattices are Hamilton-laceable. In *Proceedings of the Ninth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1978)*, Congress. Numer., XXI, pages 649–661. Utilitas Math., Winnipeg, Man., 1978.

36 P. J. Slater. Generating all permutations by graphical transpositions. *Ars Combin.*, 5:219–225, 1978.

37 G. Stachowiak. Hamilton paths in graphs of linear extensions for unions of posets. *SIAM J. Discrete Math.*, 5(2):199–206, 1992. doi:10.1137/0405016.

38 D. Tang and C. Liu. Distance-2 cyclic chaining of constant-weight codes. *IEEE Trans. Computers*, C-22:176–180, 1973.

39 M. Tchuente. Generation of permutations by graphical exchanges. *Ars Combin.*, 14:115–122, 1982.

40 H. F. Trotter. Algorithm 115: Perm. *Commun. ACM*, 5(8):434–435, 1962. doi:10.1145/368637.368660.

41 A. Williams. The greedy Gray code algorithm. In *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, pages 525–536, 2013. doi:10.1007/978-3-642-40104-6_46.

42 P. Winkler. *Mathematical puzzles: a connoisseur's collection.* A K Peters, Ltd., Natick, MA, 2004.

43 S. Zaks. A new algorithm for generation of permutations. *BIT*, 24(2):196–204, 1984. doi:10.1007/BF01937486.

Improved Quantum Lower and Upper Bounds for Matrix Scaling

Sander Gribling
IRIF, Université de Paris, CNRS, France

Harold Nieuwboer
Korteweg–de Vries Institute for Mathematics and QuSoft,
University of Amsterdam, The Netherlands

Abstract

Matrix scaling is a simple to state, yet widely applicable linear-algebraic problem: the goal is to scale the rows and columns of a given non-negative matrix such that the rescaled matrix has prescribed row and column sums. Motivated by recent results on first-order quantum algorithms for matrix scaling, we investigate the possibilities for quantum speedups for classical second-order algorithms, which comprise the state-of-the-art in the classical setting.

We first show that there can be essentially no quantum speedup in terms of the input size in the high-precision regime: any quantum algorithm that solves the matrix scaling problem for $n \times n$ matrices with at most m non-zero entries and with ℓ_2-error $\varepsilon = \widetilde{\Theta}(1/m)$ must make $\widetilde{\Omega}(m)$ queries to the matrix, even when the success probability is exponentially small in n. Additionally, we show that for $\varepsilon \in [1/n, 1/2]$, any quantum algorithm capable of producing $\frac{\varepsilon}{100}$-ℓ_1-approximations of the row-sum vector of a (dense) normalized matrix uses $\Omega(n/\varepsilon)$ queries, and that there exists a constant $\varepsilon_0 > 0$ for which this problem takes $\Omega(n^{1.5})$ queries.

To complement these results we give improved quantum algorithms in the low-precision regime: with quantum graph sparsification and amplitude estimation, a box-constrained Newton method can be sped up in the large-ε regime, and outperforms previous quantum algorithms. For entrywise-positive matrices, we find an ε-ℓ_1-scaling in time $\widetilde{O}(n^{1.5}/\varepsilon^2)$, whereas the best previously known bounds were $\widetilde{O}(n^2 \mathrm{polylog}(1/\varepsilon))$ (classical) and $\widetilde{O}(n^{1.5}/\varepsilon^3)$ (quantum).

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Quantum computation theory

Keywords and phrases Matrix scaling, quantum algorithms, lower bounds

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.35

Related Version *Full Version*: `https://arxiv.org/abs/2109.15282v1` [17]

Funding *Sander Gribling*: Partially supported by SIRTEQ-grant QuIPP.
Harold Nieuwboer: Supported by NWO grant OCENW.KLEIN.267.

Acknowledgements We thank Joran van Apeldoorn, Michael Walter and Ronald de Wolf for interesting and helpful discussions, and the latter two for giving feedback on a first version of this paper. Moreover, we thank Ronald de Wolf for pointing us to [26], which allows for an exponentially small success probability in Theorem 2.1.

1 Introduction

The matrix scaling problem asks to scale each row and column of a given matrix $\mathbf{A} \in [0,1]^{n \times n}$ by a positive number in such a way that the resulting matrix has marginals (i.e., row- and column-sums) that are close to some prescribed marginals. For example, one could ask to scale the matrix in such a way that it becomes doubly stochastic.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 35; pp. 35:1–35:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Matrix scaling has applications in a wide variety of areas including numerical linear algebra [4], optimal transport in machine learning [13], statistics [23, 14, 9, 8], and also in more theoretical settings, e.g. for approximating the permanent [28]. For a survey, we refer the reader to [19]. Furthermore, the matrix scaling problem is a special (commutative) instance of a more general (non-commutative) class of problems, which includes operator and tensor scaling; these problems have many more applications and are a topic of much recent interest [16, 10].

Formally, the matrix scaling problem is defined for the ℓ_p-norm as follows. Given a matrix $\mathbf{A} \in [0,1]^{n\times n}$ with at most m non-zero entries, entrywise-positive target marginals $\mathbf{r}, \mathbf{c} \in \mathbb{R}^n$ with $\|\mathbf{r}\|_1 = 1 = \|\mathbf{c}\|_1$, and a parameter $\varepsilon \geq 0$, find vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ such that the (rescaled) matrix $\mathbf{A}(\mathbf{x},\mathbf{y}) := (A_{ij}e^{x_i+y_j})_{i,j\in[n]}$ satisfies

$$\|\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y})) - \mathbf{r}\|_p \leq \varepsilon, \qquad \|\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y})) - \mathbf{c}\|_p \leq \varepsilon. \tag{1.1}$$

Here $\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y})) = (\sum_{j=1}^n A_{ij}e^{x_i+y_j})_{i\in[n]}$ is the vector of row-marginals of the matrix $\mathbf{A}(\mathbf{x},\mathbf{y})$ and similarly $\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y})) = (\sum_{i=1}^n A_{ij}e^{x_i+y_j})_{j\in[n]}$ is the vector of column-marginals. We refer to $\mathbf{x}$ and $\mathbf{y}$ as the scaling vectors, whereas e^{x_i} and e^{y_j} are called scaling factors. A common choice of target marginals is $(\mathbf{r},\mathbf{c}) = (\frac{\mathbf{1}}{n}, \frac{\mathbf{1}}{n})$, i.e., every row and column sum target is $1/n$, and we refer to these as the *uniform* target marginals. As is standard in the matrix scaling literature, we will henceforth assume that $\mathbf{A}$ is *asymptotically* $(\mathbf{r},\mathbf{c})$*-scalable*: for every $\varepsilon > 0$, there exist $\mathbf{x},\mathbf{y}$ such that $\mathbf{A}(\mathbf{x},\mathbf{y})$ satisfies Equation (1.1). This depends only on the support of $\mathbf{A}$ [30, Thm. 3], and is the case if and only if $(\mathbf{r},\mathbf{c})$ is in the convex hull of the points $(\mathbf{e}_i,\mathbf{e}_j) \in \mathbb{R}^{2n}$ such that $A_{ij} > 0$, where the $\mathbf{e}_i$ are the standard basis vectors for $\mathbb{R}^n$. We will also always assume the smallest non-zero entry of each of $\mathbf{A}$, $\mathbf{r}$ and $\mathbf{c}$ is at least $1/\mathrm{poly}(n)$.

Many classical algorithms for the matrix scaling problem can be viewed from the perspective of convex optimization. For example, one can solve the matrix scaling problem by minimizing the convex (potential) function

$$f(\mathbf{x},\mathbf{y}) = \sum_{i,j=1}^{n} A_{ij}e^{x_i+y_j} - \langle \mathbf{r},\mathbf{x}\rangle - \langle \mathbf{c},\mathbf{y}\rangle, \tag{1.2}$$

where $\langle\cdot,\cdot\rangle$ denotes the standard inner product on $\mathbb{R}^n$. The popular and practical Sinkhorn algorithm [31] – which alternates between rescaling the rows and columns to the desired marginals – can be viewed as a (block-)coordinate descent algorithm on f, i.e., a first-order method. Given its simplicity, it is no wonder that it has been rediscovered in many settings, and is known by many names, such as the RAS algorithm, iterative proportional fitting, or raking.

It is known that the iterates in the Sinkhorn algorithm converge to a $(\mathbf{r},\mathbf{c})$-scaled matrix whenever $\mathbf{A}$ is asymptotically $(\mathbf{r},\mathbf{c})$-scalable. The convergence rate of Sinkhorn's algorithm is known in various settings, and we give a brief overview of the (classical) time complexity of finding an ε-ℓ_1-scaling, noting that a single iteration can be implemented in time $\widetilde{O}(m)$. When $\mathbf{A}$ is entrywise positive then one can scale in time $\widetilde{O}(n^2/\varepsilon)$ [15]; in the ℓ_2-setting for uniform target marginals a similar result can be found in [21, 20]. In the general setting where $\mathbf{A}$ has at most $m \leq n^2$ non-zero entries the complexity becomes $\widetilde{O}(m/\varepsilon^2)$ (for arbitrary target marginals $(\mathbf{r},\mathbf{c})$); a proof may be found in [2] for the entrywise-positive case, [11] for exactly scalable matrices (i.e., where the problem can be solved for $\varepsilon = 0$) and [5] for asymptotically scalable matrices.

While simple, the Sinkhorn algorithm is by no means the fastest when the parameter ε is small. The classical state-of-the-art algorithms are based on second-order methods such as (traditional) interior point methods or so-called *box-constrained Newton methods* [12, 1], the latter of which we describe in more detail below. We note that these algorithms depend on fast algorithms for graph sparsification and Laplacian system solving, so are rather complicated compared to Sinkhorn's algorithm. The box-constrained Newton methods can find ε-ℓ_1-scaling vectors in time $\widetilde{O}(mR_\infty)$, where the $\widetilde{O}$ hides polylogarithmic factors in n and $1/\varepsilon$, and R_∞ is a certain diameter bound (made precise later in the introduction). For entrywise-positive matrices, R_∞ is of size $\widetilde{O}(1)$, and in general it is known to be $\widetilde{O}(n)$ [1, Lem. 3.3]. Alternatively, the interior-point method of [12] has a time complexity of $\widetilde{O}(m^{3/2})$, which is better than the box-constrained Newton method for general inputs, but worse for entrywise-positive matrices.

Recently, a quantum algorithm for matrix scaling was developed based on Sinkhorn's algorithm [5]. It uses quantum approximate counting for computing marginals, and finds ε-ℓ_1-scaling vectors in time $\widetilde{O}(\sqrt{mn}/\varepsilon^4)$ for general matrices or $\widetilde{O}(n^{1.5}/\varepsilon^3)$ for entrywise-positive matrices. This improves the dependence on m and n at the cost of a higher dependence on $1/\varepsilon$ when compared to the classical Sinkhorn algorithm (which we recall runs in $\widetilde{O}(m/\varepsilon^2)$, or $\widetilde{O}(n^2/\varepsilon)$ for entrywise-positive matrices). Furthermore, it was shown that this quantum algorithm is optimal for (sufficiently small) constant ε: there exists an $\varepsilon_0 > 0$ (independent of n) such that every quantum algorithm that ε_0-ℓ_1-scales to uniform target marginals with probability at least $2/3$ must make at least $\Omega(\sqrt{mn})$ queries. It was left as an open problem whether one can also obtain quantum speedups (in terms of n or m) using second-order methods. In this work we give improved quantum lower and upper bounds on the complexity of matrix scaling. We first prove a lower bound: we show that every quantum algorithm that solves the matrix scaling problem for small enough ε must make a number of queries proportional to the number of non-zero entries in the matrix, even when the success probability of the algorithm is only assumed to be exponentially small. This shows that one cannot hope to get a quantum algorithm for matrix scaling with a polylogarithmic $1/\varepsilon$-dependence and sublinear dependence on m. However, this does not rule out that second-order methods can be useful in the quantum setting. Indeed, we give a quantum box-constrained Newton method which has a better $1/\varepsilon$-dependence than the previously mentioned quantum Sinkhorn algorithm, and in certain settings is strictly better, such as for entrywise-positive instances.

1.1 Lower bounds

As previously mentioned, we show for entrywise-positive instances that a polynomial $1/\varepsilon$-dependence is necessary for a scaling algorithm whose n-dependence is $n^{2-\gamma}$ for a constant $\gamma > 0$. More precisely, we prove the following theorem (which we extend to an $\widetilde{\Omega}(m)$-lower bound in the general setting of $m \leq n^2$ non-zero entries in Corollary 2.11):

▶ **Theorem 1.1.** *There exists a constant $C > 0$ such that every matrix scaling algorithm that, with probability $\geq \frac{3}{2}\exp(-n/100)$, finds scaling vectors for entrywise-positive $n \times n$-matrices with ℓ_2-error $C/(n^2\sqrt{\ln n})$ must make at least $\Omega(n^2)$ queries to the matrix. This even holds for uniform targets and matrices with smallest entry $\Omega(1/n^2)$.*

The proof of this lower bound is based on a reduction from deciding whether bit strings have Hamming weight $n/2 + 1$ or $n/2 - 1$. Specifically, given k bit strings $z^1, \ldots, z^k \in \{\pm 1\}^n$ for $k = \Theta(n)$, each with Hamming weight $|z^i| = n/2 + a_i$ where $a_i \in \{\pm 1\}$, we show that any matrix scaling algorithm can be used to determine all the a_i. One can show that every quantum algorithm that computes all the a_i's needs to make $\Omega(nk)$ quantum queries to the bit string $z^1, \ldots, z^k$, even if the algorithm has only exponentially small success probability: to determine a single a_i with success probability at least $2/3$, one needs to make $\Omega(n)$ quantum queries to the bit string z^i [7, 29, 3], and one can use the strong direct product theorem of Lee and Roland [26] to prove the lower bound for computing all k a_i's simultaneously. To convert the problem of computing the a_i to an instance of matrix scaling, one constructs a $2k \times n$ matrix $\mathbf{A}$ whose first k rows are (roughly) given by the vectors $1 + z^i/b$ for some $b \geq 2$, and whose last k rows are given by $1 - z^i/b$. For such an $\mathbf{A}$, the column sums are all $2k$, and the row sums are determined by the a_i. If the matrix $\mathbf{A}'$ obtained by a single Sinkhorn step from $\mathbf{A}$ (i.e., rescaling all the rows) were exactly column scaled, then the *optimal* scaling factors encode the a_i. We show that, if one randomly (independently for each i) permutes the z^i beforehand, this is approximately the case: the column sums of this $\mathbf{A}'$ will be close to the desired column sums with high probability, and hence the first step of Sinkhorn gives approximately optimal scaling factors (which encode the a_i). Then, we give a lower bound on the strong convexity parameter of the potential f, to show that *all* sufficiently precise minimizers of f also encode the a_i. In other words, from sufficiently precise scaling factors, we can recover the a_i, yielding the reduction to matrix scaling, and consequently a lower bound for the matrix scaling problem.

We additionally study the problem of computing an ε-ℓ_1-approximation of the vector of row sums of an ℓ_1-normalized $n \times n$ matrix $\mathbf{A}$. This is a common subroutine for matrix scaling algorithms; for instance, the gradient of the potential function f from (1.2) that we optimize for the upper bound can be determined from the row and column sums by subtracting the desired row and column sums, so the complexity of this subroutine directly relates to the complexity of each iteration in our algorithm. We give the following lower bound for this problem.

▶ **Theorem 1.2** (Informal). *For $\varepsilon \in [1/n, 1/2]$ and an ℓ_1-normalized matrix $\mathbf{A} \in [0,1]^{n \times n}$, computing an $\frac{\varepsilon}{100}$-ℓ_1-approximation of $\mathbf{r}(\mathbf{A})$ takes $\Omega(n/\varepsilon)$ queries to $\mathbf{A}$. Moreover, there exists a constant $\varepsilon_0 > 0$ such that computing an ε_0-ℓ_1-approximation of $\mathbf{r}(\mathbf{A})$ takes $\Omega(n^{1.5})$ queries to $\mathbf{A}$.*

The first lower bound in the theorem is proven in Theorem 2.12. Its proof is based on a reduction from $\Theta(n)$ independent instances of the majority problem, as for the lower bound for matrix scaling. The second lower bound can be derived from the lower bound for matrix scaling given in [5]: using a constant number of calls to a subroutine that provides constant-precision approximations to the row- and column-sum vectors, one can implement Sinkhorn's algorithm to find a constant-precision ℓ_1-scaling, which for a small enough constant takes $\Omega(n^{1.5})$ queries. Hence, there exists a constant $\varepsilon_0 > 0$ (independent of n) such that computing an ε_0-ℓ_1-approximation of $\mathbf{r}(\mathbf{A})$ takes at least $\Omega(n^{1.5})$ queries to the matrix entries.

1.2 Upper bounds

While the first lower bound (Theorem 1.1) shows that a (quantum) algorithm for matrix scaling cannot have both an $m^{1-\gamma}$-dependence for $\gamma > 0$ and a polylogarithmic $1/\varepsilon$-dependence, one can still hope to obtain a second-order $\widetilde{O}(\sqrt{mn}/\mathrm{poly}(\varepsilon))$-time algorithm with a better $1/\varepsilon$-dependence than the quantum Sinkhorn algorithm of [5] (which we recall is based on quantum approximate counting). We show that one can build on a box-constrained Newton method [12, 1] to obtain a quantum algorithm which achieves this, at the cost of depending quadratically on a certain diameter bound R_∞; recall for comparison that the classical box-constrained Newton methods run in time $\widetilde{O}(mR_\infty)$. For general matrices, one has the bound $R_\infty = \widetilde{O}(n)$ [1, Lem. 3.3]. The performance of the resulting quantum box-constrained Newton method is summarized in the following theorem:

▶ **Theorem 1.3** (Informal version of Corollaries 3.14 and 3.15)**.** *For asymptotically-scalable matrices $\mathbf{A} \in \mathbb{R}^{n\times n}_{\geq 0}$ with m non-zero entries and target marginals $(\mathbf{r}, \mathbf{c})$, one can find $(\mathbf{x}, \mathbf{y})$ such that $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is $O(\varepsilon)$-ℓ_1-scaled to $(\mathbf{r}, \mathbf{c})$ in quantum time $\widetilde{O}\big(R_\infty^2\sqrt{mn}/\varepsilon^2\big)$ where R_∞ is the ℓ_∞-norm of at least one ε^2-minimizer of f. When $\mathbf{A}$ is entrywise positive we have $R_\infty = \widetilde{O}(1)$, so the algorithm runs in quantum time $\widetilde{O}\big(n^{1.5}/\varepsilon^2\big)$.*

We emphasize that the diameter bound R_∞ does not need to be provided as an input to the algorithm. Note that for entrywise-positive matrices, the algorithm improves over the quantum Sinkhorn method, which runs in time $\widetilde{O}\big(n^{1.5}/\varepsilon^3\big)$.

Let us give a sketch of the box-constrained method that we use, see Section 3.1 for details. The algorithm aims to minimize the (highly structured) convex potential function f from Equation (1.2). A natural iterative method for minimizing convex functions f is to minimize in each iteration i the quadratic Taylor expansion $\frac{1}{2}\mathbf{x}^T\nabla^2 f(\mathbf{x}^{(i)})\mathbf{x} + \mathbf{x}^T\nabla f(\mathbf{x}^{(i)}) + f(\mathbf{x}^{(i)})$ of the function at the current iterate. A box-constrained method constrains the minimization of the quadratic Taylor expansion to those $\mathbf{x}$ that lie in an ℓ_∞-ball of radius c around the current iterate (hence the name):

$$\mathbf{x}^{(i)} = \operatorname*{argmin}_{\|\mathbf{x}-\mathbf{x}^{(i)}\|_\infty \leq c} \frac{1}{2}\mathbf{x}^T\nabla^2 f(\mathbf{x}^{(i)})\mathbf{x} + \mathbf{x}^T\nabla f(\mathbf{x}^{(i)}).$$

This is guaranteed to decrease a convex function f whenever it is *second-order robust*, i.e., whenever the Hessian of f at a point is a good multiplicative approximation of the Hessian at every other point in a constant-radius ℓ_∞-ball. One can show that the steps taken decrease the potential gap by a multiplicative factor which depends on the distance to the minimizer.

One then observes that the function f from Equation (1.2) is second-order robust. Moreover, its Hessian has an exceptionally nice structure: given by

$$\nabla^2 f(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \operatorname{diag}(\mathbf{r}(\mathbf{A}(\mathbf{x}, \mathbf{y}))) & \mathbf{A}(\mathbf{x}, \mathbf{y}) \\ \mathbf{A}(\mathbf{x}, \mathbf{y})^T & \operatorname{diag}(\mathbf{c}(\mathbf{A}(\mathbf{x}, \mathbf{y}))) \end{bmatrix},$$

it is similar to a *Laplacian* matrix. This means that the key subroutine in this method (approximately) minimizes quadratic forms $\frac{1}{2}\mathbf{z}^T\mathbf{H}\mathbf{z} + \mathbf{z}^T\mathbf{b}$ over ℓ_∞-balls, where $\mathbf{H}$ is a Laplacian matrix; without the ℓ_∞-constraint, this amounts to solving the Laplacian system $\mathbf{H}\mathbf{z} = \mathbf{b}$. Such a subroutine can be implemented for the more general class of symmetric diagonally-dominant matrices (with non-positive off-diagonal entries) on a classical computer in (almost) linear time in the number of non-zero entries of $\mathbf{H}$ [12]. For technical reasons, one has to add a regularization term to f, and the regularized potential instead has a symmetric diagonally-dominant Hessian structure. Given the recent quantum algorithm for graph sparsification and Laplacian system solving of Apers and de Wolf [6], one would therefore hope to obtain a quantum speedup for the box-constrained Newton method. We show that one can indeed achieve this by first using the quantum algorithm for graph sparsification, and then using the classical method for the minimization procedure. We note, however, that in order to achieve a quantum speedup in terms of m and n, we incur a polynomial dependence in the time complexity on the precision with which we can approximate $\mathbf{H}$ and $\mathbf{b}$ (as opposed to only a *polylogarithmic* dependence classically). Such a speedup with respect to one parameter (dimension) at the cost of a slowdown with respect to another (precision) is more common in recent quantum algorithms for optimization problems and typically requires a more careful analysis of the impact of approximation errors. Interestingly, for the classical box-constrained Newton method, the minimization subroutine is the bottleneck, whereas in our quantum algorithm, the cost of a single iteration is dominated by the time it takes to approximate the vector $\mathbf{b}$. Using quantum approximate counting (carefully) as in [5], one can obtain an additive $\delta \cdot \|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1$-approximation of $\mathbf{b}$ in time roughly $\sqrt{mn}/\delta$. To obtain an efficient quantum algorithm we therefore need to control $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1$ throughout the run of the algorithm. We do so efficiently by testing in each iteration whether the 1-norm of $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is too large, if it is, we divide the matrix by 2 (by shifting $\mathbf{x}$ by an appropriate multiple of the all-ones vector), which reduces the potential.

1.3 Open problems

Our lower bound on matrix scaling shows that it is not possible to provide significant quantum speedups for scaling of entrywise-positive matrices in the high-precision scaling regime. However, the best classical upper bound for ε-scaling when no assumptions are made on the support of the matrices is $\widetilde{O}(m^{3/2})$, where m is the number of non-zero entries [12] (recall that this hides a polylogarithmic dependence on $1/\varepsilon$). The algorithm that achieves this bound is an interior-point method, rather than a box-constrained Newton method. It is an interesting open problem whether such an algorithm also admits a quantum speedup in terms of m while retaining a polylogarithmic $1/\varepsilon$-dependence. Note that while the interior-point method relies on fast Laplacian system solvers, it is not enough to merely replace this by a quantum Laplacian system solver, as the dimension of the linear system in question is $m + n$ rather than $\Theta(n)$. More generally, the possibility of obtaining quantum advantages in high-precision regimes for optimization problems is still a topic of ongoing investigation.

A second natural question is whether the lower bounds from Theorem 1.2 for computing an approximation of the row sums are tight. The best upper bound for the row-sum vector approximation that we are aware of is the one we use in our scaling algorithm: we can compute an ε-ℓ_1-approximation of the row sums in time $\widetilde{O}(n^{1.5}/\varepsilon)$. For constant $\varepsilon_0 \geq \varepsilon > 0$ this matches the lower bound $\Omega(n^{1.5})$ (up to log-factors), but for non-constant $\varepsilon > \frac{1}{100n}$ it remains an interesting open problem to close the gap between $\widetilde{O}(n^{1.5}/\varepsilon)$ and $\Omega(n/\varepsilon)$.

2 Lower bounds for matrix scaling and marginal approximation

In this section we prove two lower bounds: an $\widetilde{\Omega}(m)$-lower bound for $1/\mathrm{poly}(n)$-ℓ_2-scaling $n \times n$ matrices with at most m non-zero entries, and for $\varepsilon \in [1/n, 1/2]$ an $\Omega(n/\varepsilon)$-lower bound for ε-ℓ_1-approximation of the row-sum vector of a normalized $n \times n$ matrix (with non-negative entries). The proofs for both lower bounds are based on a reduction from the lower bound given below in Theorem 2.1. In Section 2.1 we construct the associated instances for matrix scaling, and in Section 2.2 we analyze their column marginals after a single iteration of the Sinkhorn algorithm. Afterwards, in Section 2.3 we show that these column marginals are close enough to the target marginals for the reduction to matrix scaling to work, and in Section 2.4 we put the ingredients together, with the main theorem being Theorem 2.10. Finally, in Section 2.5 we prove the lower bound for computing approximations to the row marginals. The lower bound we reduce from is the following:

▶ **Theorem 2.1.** *Let n be even, $\tau \in [1/n, 1/2]$ such that $n\tau$ is an integer, and let $k \geq 1$ be an integer. Given k binary strings $z^1, \dots, z^k \in \{\pm 1\}^n$, where z^i has Hamming weight $n/2 + a_i \tau n$ for $a_i \in \{-1, 1\}$, computing with probability $\geq \exp(-k/100)$ a string $\tilde{a} \in \{-1, 1\}^k$ that agrees with a in $\geq 99\%$ of the positions requires $\Omega(k/\tau)$ quantum queries.*

Proof. Let $\mathcal{D} = \{z \in \{\pm 1\}^n : |z| = n/2 + \tau n \text{ or } |z| = n/2 - \tau n\}$ and define the partial Boolean function $f \colon \mathcal{D} \to \{\pm 1\}$ by $f(z) = 1$ if $|z| = n/2 + \tau n$, and $f(z) = -1$ otherwise. It is known that computing f with probability at least $2/3$ takes $\Theta(1/\tau)$ quantum queries to z [29, Cor. 1.2], i.e., the bounded-error quantum query complexity $Q_{1/3}(f)$ is $\Theta(1/\tau)$.

We now proceed with bounding the query complexity of computing 99% of the entries of $f^{(k)} \colon \mathcal{D}^k \to \{\pm 1\}^k$ defined by $f^{(k)}(z^1, \dots, z^k) = (f(z^1), \dots, f(z^k))$. We will make use of the general adversary bound $\mathrm{Adv}^{\pm}(f)$ [18] which is known to satisfy $\mathrm{Adv}^{\pm}(f) = \Theta(Q_{1/3}(f))$ [25, Thm. 1.1]. The strong direct product theorem of Lee and Roland [26, Thm. 5.5] says that for every $0 \le \delta < 1$, $\mu \in [\frac{1+\sqrt{\delta}}{2}, 1]$ and integers k, K, every quantum algorithm that outputs a bit string $\tilde{a} \in \{\pm 1\}^k$, and makes T quantum queries to the bit strings $z^1, \dots, z^k$ with $T \le \frac{k\delta}{K(1-\delta)} \mathrm{Adv}^{\pm}(f)$ has the property that $\tilde{a}$ agrees with $f^{(k)}(z^1, \dots, z^k)$ on at least a μ-fraction of the entries with probability at most $\exp(k(\frac{1}{K} - D(\mu \| \frac{1+\sqrt{\delta}}{2})))$.[1] Here $D(\mu \| \frac{1+\sqrt{\delta}}{2})$ is the Kullback–Leibler divergence between the distributions $(\mu, 1-\mu)$ and $(\frac{1+\sqrt{\delta}}{2}, \frac{1-\sqrt{\delta}}{2})$. For $\mu = 0.99$, $\delta = 0.1$ and $K = 3$, one has $\frac{1}{K} - D(\mu \| \frac{1+\sqrt{\delta}}{2}) \approx -0.03 \le -1/100$. Therefore, the strong direct product theorem shows that computing 99% of the entries of $f^{(k)}(z^1, \dots, z^k) = a$ correctly, with success probability at least $\exp(-k/100)$, takes $\Omega(k\, \mathrm{Adv}^{\pm}(f)) = \Omega(k\, Q_{1/3}(f)) = \Omega(k/\tau)$ quantum queries. ◀

We will use this lower bound with $k = n/2$ and $\tau = 1/n$. The following intuition is useful to keep in mind. For a fixed $b \ge 2$, define the $2k \times n$ matrix $\mathbf{A}$ whose $(2i-1)$-th row equals $1 + z^i/b$ and whose $(2i)$-th row equals $1 - z^i/b$. Then $\mathbf{A}$ has the property that the row-marginals encode the Hamming weights of the z^i, and are all very close to n. (This implies that the first row-rescaling step of Sinkhorn's algorithm encodes the a_i.) Moreover, the column-marginals are exactly uniform. Hence, one may hope that all sufficiently precise scalings of $\mathbf{A}$ to uniform targets have scaling factors that are close to those given by the first row-rescaling step of Sinkhorn's algorithm (and hence learn most of the a_i).

Below we formalize this approach. We show that if one randomly permutes the coordinates of each z^i (independently over i), then with high probability, all ε-scalings of the resulting matrix $\mathbf{A}^{\sigma}$ are close to the first step of Sinkhorn's algorithm; here we need to choose b sufficiently large ($\sim \sqrt{\ln(n)}$) and ε sufficiently small ($\sim \frac{1}{n^2 b}$). The section is organized as follows. In Section 2.1 we formally define our matrix scaling instances and we analyse the first row-rescaling step of Sinkhorn's algorithm. In Section 2.2 we show that after the row-rescaling step, with high probability (over the choice of permutations), the column-marginals are close to uniform. In Sections 2.3 and 2.4 we use the strong convexity of the potential f from Equation (1.2) to show that if the above event holds, then all approximate minimizers of f can be used to solve the counting problem.

2.1 Definition of the scaling instances and analysis of row marginals

Let $n \ge 4$ be even. Let $k = n/2$ and let $z^1, \dots, z^k \in \{\pm 1\}^n$ have Hamming weight $|z^i| = |\{j : z^i_j = 1\}| = n/2 + a_i$ for $a_i \in \{\pm 1\}$. Sample uniformly random permutations $\sigma^1, \dots, \sigma^k \in S_n$ and define w^i by $w^i_j = z^i_{(\sigma^i)^{-1}(j)}$. Let $b \ge 2$ be some number depending on n, and consider the $2k \times n$ matrix $\mathbf{A}^{\sigma}$ whose entries are $\mathbf{A}^{\sigma}_{2i-1,j} = 1 + \frac{w^i_j}{b}$ and $\mathbf{A}^{\sigma}_{2i,j} = 1 - \frac{w^i_j}{b}$. Then each column sum $c_j(\mathbf{A}^{\sigma})$ is $2k$, and the row sums of $\mathbf{A}^{\sigma}$ are given by

$$r_{2i-1}(\mathbf{A}^{\sigma}) = n + \frac{1}{b}\sum_{j=1}^{n} w^i_j = n + \frac{2}{b} a_i, \quad r_{2i}(A^{\sigma}) = n - \frac{2}{b} a_i.$$

[1] In [26] the upper bound on T is stated in terms of $\mathrm{Adv}^*(F)$ where $F = (\delta_{f(x),f(y)})_{x,y \in \mathcal{D}}$ is the Gram matrix of f. For Boolean functions f one has $\mathrm{Adv}^*(F) = \mathrm{Adv}^{\pm}(f)$ [25, Thm. 3.4].

Let

$$X_{2i-1} = \frac{1}{2k} \cdot \frac{1}{n + \frac{2}{b} a_i} \text{ and } X_{2i} = \frac{1}{2k} \cdot \frac{1}{n - \frac{2}{b} a_i} \qquad \text{for all } i \in [k] \tag{2.1}$$

be the row scaling factors obtained from a single Sinkhorn step. We first observe that the difference between $x_{2i-1} := \ln(X_{2i-1})$ and $x_{2i} := \ln(X_{2i})$ permits to recover a_i.

▶ **Lemma 2.2.** *For the specific row-scaling factors* $\mathbf{X}$ *for* $\mathbf{A}^\sigma$ *given in* (2.1)*, for every* $i \in [k]$ *it holds that* $|\ln(X_{2i-1}/X_{2i})| \geq \frac{4}{nb}$*, and* $\operatorname{sign}(\ln(X_{2i}/X_{2i-1})) = a_i$*.*

Proof. Using $nb > 2$, we have $|\ln(X_{2i-1}/X_{2i})| = \left|\ln\left(\frac{n+\frac{2}{b}}{n-\frac{2}{b}}\right)\right| = \ln\left(\frac{nb+2}{nb-2}\right) \geq \frac{4}{nb}$. ◀

2.2 Concentration of column marginals

We record here an explicit expression for the j-th column marginal of $\mathbf{X}\mathbf{A}^\sigma$ for the $\mathbf{X}$ from (2.1), which follows from straightforward algebraic manipulations.

▶ **Lemma 2.3.** *We have* $c_j(\mathbf{X}\mathbf{A}^\sigma) = \frac{1}{2k(n^2-4/b^2)}\left(2kn - \frac{4}{b^2}\sum_{i=1}^k w_j^i a_i\right)$ *for* $j \in [n]$*.*

We now show that with high probability (over the choice of permutations) the column marginals are close to uniform. To do so, we first compute the expectation of $\sum_{i=1}^k w_j^i a_i$ (Corollary 2.5). This quantity allows us to obtain the desired concentration of the column marginals via Hoeffding's inequality (Lemma 2.6).

▶ **Lemma 2.4.** *Let* $I = \{i \in [k] : a_i = 1\}$ *and* $I^c = [k] \setminus I$*. Define random variables* W_j*,* W_j^c *by* $W_j = \sum_{i \in I} w_j^i$ *and* $W_j^c = \sum_{i \in I^c} w_j^i$*. Then* $\mathbb{E}[W_j] = \frac{2|I|}{n}$ *and* $\mathbb{E}[W_j^c] = -\frac{2|I^c|}{n}$*.*

Proof. Observe that each w_j^i is 1 with probability $\frac{1}{2} + \frac{a_i}{n}$ because σ^i is chosen uniformly randomly from S_n, and is -1 with probability $\frac{1}{2} - \frac{a_i}{n}$. Therefore $\mathbb{E}[w_j^i] = \frac{2a_i}{n}$. By linearity of expectation, the result follows. ◀

▶ **Corollary 2.5.** *We have* $\mathbb{E}\left[\sum_{i=1}^k w_j^i a_i\right] = \mathbb{E}[W_j] - \mathbb{E}[W_j^c] = \frac{2(|I|+|I^c|)}{n} = \frac{2k}{n}$*.*

▶ **Lemma 2.6.** *For* $t \geq 0$ *and* $j \in [n]$*, with probability at least* $1 - 2e^{-t^2/2}$*, we have* $\left|c_j(\mathbf{X}\mathbf{A}^\sigma) - \frac{1}{n}\right| = O\left(\frac{t}{b^2 n^2 \sqrt{k}}\right)$*.*

Proof. One can verify that $\left|c_j(\mathbf{X}\mathbf{A}^\sigma) - \frac{1}{n}\right| = \frac{4}{2kn(n^2-4/b^2)b^2}\left|2k - n\sum_{i=1}^k w_j^i a_i\right|$. For fixed j and distinct $i, i' \in [k]$, w_j^i and $w_j^{i'}$ are independently distributed random variables because σ^i and $\sigma^{i'}$ are independent. Therefore, $V_j := W_j - W_j^c = \sum_{i=1}^k w_j^i a_i$ is a sum of k independent random variables, with each $a_i w_j^i \in [-1, 1]$, and Hoeffding's inequality yields for any $t \geq 0$ that $\Pr[|V_j - \mathbb{E}[V_j]| \geq t \cdot \sqrt{k}] \leq 2\exp(-t^2/2)$. Assuming that $|V_j - \mathbb{E}[V_j]| \leq t\sqrt{k}$, we have $\left|2k - n\sum_{i=1}^k a_i w_j^i\right| = n|\mathbb{E}[V_j] - V_j| \leq nt\sqrt{k}$. With this estimate, we see that

$$\left|c_j(\mathbf{X}\mathbf{A}^\sigma) - \frac{1}{n}\right| \leq \frac{4}{2kn(n^2-4/b^2)b^2} \cdot nt\sqrt{k} = \frac{2t}{b^2(n^2-4/b^2)\sqrt{k}}.$$

◀

▶ **Corollary 2.7.** *For any* $t \geq 0$*, with probability* $\geq 1 - 2ne^{-t^2/2}$*, we have* $\left\|c(\mathbf{X}\mathbf{A}^\sigma) - \frac{\mathbf{1}}{n}\right\|_2 \leq \frac{2\sqrt{n}t}{b^2(n^2-4/b^2)\sqrt{k}} = O\left(\frac{t}{b^2 n^2}\right)$*.*

2.3 Strong convexity properties of the potential

For a λ-strongly convex function f, the set $\{\mathbf{z} : \|\nabla f(\mathbf{z})\|_2 \leq \varepsilon\}$ has a diameter that is bounded by a function of λ (we make this well-known fact precise in Lemma A.4). We show that our potential is strongly convex when viewed as a function from (a suitable subset of) the linear subspace $V = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n : \langle(\mathbf{x}, \mathbf{y}), (\mathbf{1}_n, -\mathbf{1}_n)\rangle = 0\}$ to $\mathbb{R}$ (note that f is invariant under translation by multiples of $(\mathbf{1}_n, -\mathbf{1}_n)$). We use this to prove the following lemma, which shows that whenever $\nabla f(\mathbf{x}, \mathbf{y})$ is small, $(\mathbf{x}, \mathbf{y})$ is close to the minimizer of f on V. It is easy to verify that Corollary 2.7 in fact gives an upper bound on the ℓ_2-norm of the gradient at $(\ln(\mathbf{X}), \mathbf{0})$ (with $\mathbf{X}$ as in (2.1)). This implies that $(\ln(\mathbf{X}), \mathbf{0})$ is close to the minimizer of f on V, and by the triangle inequality, is also close to any other $(\mathbf{x}, \mathbf{y})$ for which $\|\nabla f(\mathbf{x}, \mathbf{y})\|_2$ is small. The full proof is given in Appendix A.

▶ **Lemma 2.8.** *Let $f\colon V \subset \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the standard potential for the matrix $\mathbf{A}^\sigma$, where V is the orthogonal complement of $(\mathbf{1}_n, -\mathbf{1}_n)$. Then for every $(\mathbf{x}, \mathbf{y}) \in V$ and $\delta \in (0, 1)$, if $\|\nabla f(\mathbf{x}, \mathbf{y})\|_2 \leq \frac{\delta}{27ne^2}$, then $\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq \delta$.*

2.4 Concluding the lower bound for matrix scaling

Let $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in V$ be the unique vector such that $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - (\mathbf{x}, \mathbf{y})$ is a multiple of $(\mathbf{1}_n, -\mathbf{1}_n)$, where $(\mathbf{x}, \mathbf{y})$ are the scaling vectors of the first step of Sinkhorn. By choosing t and b appropriately we obtain, with high probability over the choice of permutations, a bound on the distance between $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and the unique scaling vectors $(\mathbf{x}^*, \mathbf{y}^*) \in V$ of an exact scaling of $\mathbf{A}^\sigma$. This allows us to conclude that, with high probability, all sufficiently precise scalings of $\mathbf{A}^\sigma$ encode the Hamming weights a_i.

▶ **Corollary 2.9.** *There exists a constant $C > 0$ such that for $b = C\sqrt{\ln n}$ the following holds. With probability $\geq 2/3$ (over the choice of σ) we have for the exact scaling vectors $(\mathbf{x}^*, \mathbf{y}^*) \in V$ of $\mathbf{A}^\sigma$ that $a_i = \operatorname{sign}(x^*_{2i} - x^*_{2i-1})$ for all i. Furthermore, there exists a constant $C' > 0$ such that for any (x', y') that yield a (C'/n^2b)-ℓ_2-scaling of $\mathbf{A}^\sigma$, a_i can be recovered from x' as $a_i = \operatorname{sign}(x_{2i} - x_{2i-1}) = \operatorname{sign}(x'_{2i} - x'_{2i-1})$.*

Proof. Applying Corollary 2.7 with $t = 10\sqrt{\ln n}$ shows that with probability at least 2/3 we have $\|\nabla f(\bar{\mathbf{x}}, \bar{\mathbf{y}})\|_2 = \|\nabla f(\mathbf{x}, \mathbf{y})\|_2 = \frac{t}{b}\frac{2\sqrt{n}}{b(n^2-4/b^2)\sqrt{k}}$. Hence, there exists a constant $C > 0$ such that for $b = Ct$ we have $\|\nabla f(\bar{\mathbf{x}}, \bar{\mathbf{y}})\|_2 \leq \frac{1}{nb}\frac{1}{27ne^2}$. Lemma 2.8 then implies that $\|(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq \frac{1}{nb}$ and hence $|(x^*_{2i-1} - x^*_{2i}) - (x_{2i-1} - x_{2i})| \leq \frac{2}{nb}$. Together with Lemma 2.2 (which shows that $|x_{2i-1} - x_{2i}| \geq \frac{4}{nb}$) this means that $a_i = \operatorname{sign}(x^*_{2i} - x^*_{2i-1})$. Moreover, $|x^*_{2i-1} - x^*_{2i}| \geq \frac{2}{nb}$.

Now consider approximate scalings of $\mathbf{A}^\sigma$. Without loss of generality we may assume that the (x', y') that yield a $(\frac{1}{2nb}\frac{1}{27ne^2})$-$\ell_2$-scaling of $\mathbf{A}^\sigma$ belong to V (otherwise we shift it by an appropriate multiple of $(\mathbf{1}_n, -\mathbf{1}_n)$). Then, again due to Lemma 2.8, we obtain that $\|(\mathbf{x}', \mathbf{y}') - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq \frac{1}{2nb} \leq \frac{1}{4}|x^*_{2i-1} - x^*_{2i}|$ and hence $|(x'_{2i-1} - x'_{2i}) - (x^*_{2i-1} - x^*_{2i})| \leq \frac{1}{2}|x^*_{2i-1} - x^*_{2i}|$ which means that $\operatorname{sign}(x'_{2i} - x'_{2i-1}) = \operatorname{sign}(x^*_{2i-1} - x^*_{2i}) = a_i$. ◀

▶ **Theorem 2.10.** *There exists a constant $C > 0$ such that any matrix scaling algorithm that, with probability $\geq \frac{3}{2}\exp(-n/100)$, finds scalings for $n \times n$-matrices with ℓ_2-error $C/(n^2\sqrt{\ln n})$ must make at least $\Omega(n^2)$ queries to the matrix. This even holds for uniform targets and entrywise-positive matrices with smallest entry $\Omega(1/n^2)$.*

Proof. We construct a set of hard instances as in Section 2.1. Let $n \geq 4$ be even. Let $k = n/2$ and let $z^1, \ldots, z^k \in \{\pm 1\}^n$ have Hamming weight $|z^i| = |\{j : z^i_j = 1\}| = n/2 + a_i$ for $a_i \in \{\pm 1\}$. By Theorem 2.1, finding at least 99% of the a_i's with probability $\geq \exp(-n/100)$ takes $\Omega(n^2)$-queries to the z^i_j. One can recover the a_i's with probability $\geq 2/3$ as follows.

First, sample the $\sigma^1, \ldots, \sigma^{n/2}$ uniformly from S_n. A single query to $\mathbf{A}^\sigma$ takes a single query to some w^i, which takes a single query to z^i. Using Corollary 2.9, there exists a constant $C > 0$ such that, with probability $\geq 2/3$, any scaling of $\mathbf{A}^\sigma$ with ℓ_2-error $C/(n^2\sqrt{\ln n})$ recovers *all* a_i's. Therefore any matrix scaling algorithm finding such a scaling with probability $\geq \exp(-n/100)$ allows us to find all a_i's with probability $\geq \exp(-n/100)$. ◀

▶ **Corollary 2.11.** *There exist constants $C_0, C_1 > 0$ such that every matrix scaling algorithm that, with probability $\geq \exp(-C_0 n/\ln(n))$, finds scalings for $n \times n$-matrices with at most m non-zero entries and ℓ_2-error $C_1/(m\sqrt{\ln(m/n)})$ must make at least $\widetilde{\Omega}(m)$ queries. This even holds for uniform targets and matrices with smallest non-zero entry $\Omega(1/m)$.*

2.5 Lower bound for computing the row marginals

In Theorem 2.12 we show that computing an ε-ℓ_1-approximation of the row (or column marginals) of an entrywise-positive $n \times n$ matrix takes $\Omega(n/\varepsilon)$ queries to its entries (for $\varepsilon = \Omega(1/n)$). As a consequence, the same holds for computing an approximation of the gradient of common (convex) potential functions used for matrix scaling – among which is the potential we use in Section 3 – takes as many queries. Although the bound does not imply that testing whether a matrix is ε-ℓ_1-scaled takes at least $\Omega(n/\varepsilon)$ queries, it gives reasonable evidence that this should be the case. The proof can be found in the full version [17].

▶ **Theorem 2.12.** *Let $\tau \in [1/n, 1/2]$. Suppose we have a quantum algorithm that, given query access to a positive $n \times n$ matrix $\mathbf{A}$ with row-sums $\mathbf{r} = (r_1, \ldots, r_n)$ and column-sums $\mathbf{c} = (1/n, \ldots, 1/n)$, outputs (with probability $\geq \exp(-n/100)$) a vector $\tilde{\mathbf{r}} \in \mathbb{R}^n_+$ such that $\|\tilde{\mathbf{r}} - \mathbf{r}\|_1 < \tau/100$. Then this algorithm uses $\Omega(n/\tau)$ queries.*

3 Quantum box-constrained Newton method for matrix scaling

In this section, we show how to obtain a quantum speedup based on the box-constrained Newton method for matrix scaling from [12], with the main result being Theorem 3.13, and its consequences for matrix scaling given in Corollaries 3.14 and 3.15. We first recall some of the concepts that are used in the algorithm, including the definition of second-order robust convex functions, the notion of a k-oracle, and a theorem regarding efficient (classical) implementation of a k-oracle for the class of symmetric diagonally-dominant matrices with non-positive off-diagonal entries. We then show that for a second-order robust function $g\colon \mathbb{R}^n \to \mathbb{R}$ and a given $\mathbf{x} \in \mathbb{R}^n$ such that the sublevel set $\{\mathbf{x}' : g(\mathbf{x}') \leq g(\mathbf{x})\}$ is bounded, one can use a k-oracle and approximations to the gradient and Hessian of g to find a vector $\mathbf{x}'$ such that the potential gap $g(\mathbf{x}') - g(\mathbf{x}^*)$ is smaller than $g(\mathbf{x}) - g(\mathbf{x}^*)$ where $\mathbf{x}^*$ is a minimizer of g. This result extends [12, Thm. 3.4] to a setting where one can only obtain rough approximations of the gradient and Hessian of g. We then show that this applies to a regularized version $\tilde{f}$ of the potential f discussed in the introduction; to approximate the Hessian of $\tilde{f}$, we use a quantum algorithm for graph sparsification, whereas we approximate the gradient of $\tilde{f}$ using quantum approximate summing. One challenge is that the quality of the gradient approximation is directly related to the 1-norm of the matrix $\mathbf{A}(\mathbf{x}, \mathbf{y})$, so we must control this throughout the algorithm, which we achieve by manually shifting $\mathbf{x}$ when the norm becomes too large, and showing that this does not increase the regularized potential under suitable circumstances.

3.1 Minimizing second-order robust convex functions

In what follows we will minimize a convex function (potential) that satisfies a certain regularity condition: its Hessian can be approximated well on an infinity-norm ball.

▶ **Definition 3.1** ([12, Def. 3.1]). *A convex function $g\colon \mathbb{R}^n \to \mathbb{R}$ is called* second-order robust *with respect to ℓ_∞ if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with $\|\mathbf{x}-\mathbf{y}\|_\infty \leq 1$, $\frac{1}{e^2}\nabla^2 g(\mathbf{x}) \preceq \nabla^2 g(\mathbf{y}) \preceq e^2 \nabla^2 g(\mathbf{x})$.*

This implies that the local quadratic approximation to g has a good quality on a small ℓ_∞-norm ball. It is therefore natural to consider the problem of minimizing a convex quadratic function over an ℓ_∞-norm ball. We will use the following notion.

▶ **Definition 3.2** (k-oracle). *An algorithm $\mathcal{A}$ is called a* k-oracle *for a class of matrices $\mathcal{M} \subseteq \mathbb{R}^{n\times n}$ if for input $(\mathbf{H}, \mathbf{b})$ with $\mathbf{H} \in \mathcal{M}$, $\mathbf{b} \in \mathbb{R}^n$, it returns a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\|x\|_\infty \leq k$ and $\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \langle \mathbf{b}, \mathbf{x}\rangle \leq \frac{1}{2} \cdot \min_{\|\mathbf{z}\|_\infty \leq 1}(\frac{1}{2}\mathbf{z}^T\mathbf{H}\mathbf{z} + \langle \mathbf{b}, \mathbf{z}\rangle)$.*

▶ **Definition 3.3** (SDD matrix). *A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is called* symmetric diagonally-dominant *if it is symmetric, and for every $i \in [n]$, one has $A_{ii} \geq \sum_{j\neq i}|A_{ij}|$.*

In [12] it is shown how to efficiently implement an $O(\log(n))$-oracle for the class of SDD matrices $\mathbf{H}$ whose off-diagonal entries are non-positive. Their algorithm uses an efficient construction of a *vertex sparsifier chain* of $\mathbf{H}$ due to [27, 24].

▶ **Theorem 3.4** ([12, Thm. 5.11]). *Given a classical description of an SDD matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ with $\widetilde{O}(m)$ non-zero entries, such that $H_{i,j} \leq 0$ for $i \neq j$, and a classical vector $\mathbf{b} \in \mathbb{R}^n$, we can find in time $\widetilde{O}(m)$ a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\|\mathbf{x}\|_\infty = O(\log n)$ and*

$$\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \langle \mathbf{b}, \mathbf{x}\rangle \leq \frac{1}{2} \cdot \min_{\|\mathbf{z}\|_\infty \leq 1}(\frac{1}{2}\mathbf{z}^T\mathbf{H}\mathbf{z} + \langle \mathbf{b}, \mathbf{z}\rangle).$$

A k-oracle $\mathcal{A}$ gives rise to an iterative method for minimizing a second-order robust function g: starting from $x_0 \in \mathbb{R}^n$, we define a sequence $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ by

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \frac{1}{k}\Delta_i, \quad \Delta_i = \mathcal{A}\left(\frac{e^2}{k^2}\mathbf{H}_i, \frac{1}{k}\mathbf{b}_i\right)$$

where $\mathbf{H}_i$ is an approximate Hessian at $\mathbf{x}^{(i)}$, and $\mathbf{b}_i$ is an approximate gradient at $\mathbf{x}^{(i)}$. The following theorem, which is an adaptation of [12, Thm. 3.4], upper bounds the progress made in each iteration. We defer its proof to Appendix B.

▶ **Theorem 3.5.** *Let $g\colon \mathbb{R}^n \to \mathbb{R}$ be a second-order robust function with respect to ℓ_∞, let $\mathbf{x} \in \mathbb{R}^n$ be a starting point, and suppose $\mathbf{x}^*$ is a minimizer of g. Assume that we are given*

1. *a vector $\mathbf{b} \in \mathbb{R}^n$ such that $\|\mathbf{b} - \nabla g(\mathbf{x})\|_1 \leq \delta$,*
2. *two SDD matrices $\mathbf{H}_m$ and $\mathbf{H}_a$ with non-positive off-diagonal entries, such that there exists $\delta_a \geq 0$ and symmetric $\mathbf{H}'_m$ and $\mathbf{H}'_a$ satisfying $\nabla^2 g(\mathbf{x}) = \mathbf{H}'_m + \mathbf{H}'_a$ and $\frac{2}{3}\mathbf{H}_m \preceq \mathbf{H}'_m \preceq \frac{4}{3}\mathbf{H}_m$, $\quad \|\mathbf{H}_a - \mathbf{H}'_a\|_1 \leq \delta_a$.*

Let $k = O(\log n)$ be such that there exists a k-oracle $\mathcal{A}$ for the class of SDD-matrices with non-positive off-diagonal entries (cf. Theorem 3.4). Then for $\mathbf{H} = \mathbf{H}_m + \mathbf{H}_a$ and $\Delta = \mathcal{A}\left(\frac{4e^2}{3k^2}\mathbf{H}, \frac{1}{k}\mathbf{b}\right)$, the vector $\mathbf{x}' = \mathbf{x} + \frac{1}{k}\Delta$ satisfies

$$g(\mathbf{x}') - g(\mathbf{x}^*) \leq \left(1 - \frac{1}{4e^4\max(kR_\infty, 1)}\right)(g(\mathbf{x}) - g(\mathbf{x}^*)) + \frac{e^2\delta_a}{k^2} + \frac{3}{2}\delta,$$

where R_∞ is the ℓ_∞-radius of the sublevel set $\{\mathbf{x}' : g(\mathbf{x}') \leq g(\mathbf{x})\}$ about $\mathbf{x}$.

3.2 A second-order robust potential for matrix scaling and its properties

Given a sparse matrix $\mathbf{A} \in \mathbb{R}^{n\times n}_{\geq 0}$, a desired error $\varepsilon > 0$, and some number $B > 0$, we consider the regularized potential function $\tilde{f}(x, y)$ given by

$$\tilde{f}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}, \mathbf{y}) + \frac{\varepsilon^2}{ne^B}\left(\sum_i (e^{x_i} + e^{-x_i}) + \sum_j (e^{y_j} + e^{-y_j})\right),$$

where f is the commonly-used potential function from Equation (1.2). In [12], the same regularization term is used, but with a different weight (since they aim for ℓ_2-scaling and we aim for ℓ_1-scaling). The following is an adaptation of [12, Lem. 4.10], see Appendix B.2.

▶ **Lemma 3.6.** *Assume $\mathbf{A}$ is asymptotically scalable, with $\|\mathbf{A}\|_1 \leq 1$, and $\mu > 0$ its smallest non-zero entry. Let $B > 0$ and $\varepsilon > 0$ be given. Then the regularized potential $\tilde{f}$ satisfies the following properties:*

1. *$\tilde{f}$ is second-order robust with respect to ℓ_∞, and its Hessian is SDD;*
2. *we have $f(\mathbf{z}) \leq \tilde{f}(\mathbf{z})$ for any $\mathbf{z} = (\mathbf{x}, \mathbf{y})$,*
3. *for all $\mathbf{z}$ such that $\tilde{f}(\mathbf{z}) \leq \tilde{f}(\mathbf{0})$, we have $\|\mathbf{z}\|_\infty \leq B + \ln(4n + (n\ln(1/\mu)/\varepsilon^2))$, and*
4. *for any $\mathbf{z}_\varepsilon$ such that $f(\mathbf{z}_\varepsilon) \leq f^* + \varepsilon^2$ and $\|\mathbf{z}_\varepsilon\|_\infty \leq B$, one has $\tilde{f}(\mathbf{z}_\varepsilon) \leq f^* + 5\varepsilon^2$. In particular, if such a $\mathbf{z}_\varepsilon$ exists, then $|f^* - \tilde{f}^*| \leq 5\varepsilon^2$.*

In order to use Theorem 3.5 to minimize f, we need to show how to approximate both the gradient and Hessian of $\tilde{f}$. We first consider the Hessian of $\tilde{f}$, which can be written as the sum of the Hessian of f and the Hessian of the regularizer $\tilde{f} - f$. We have

$$\begin{aligned}\nabla^2 f(\mathbf{x}, \mathbf{y}) &= \begin{bmatrix}\operatorname{diag}(\mathbf{r}(\mathbf{A}(\mathbf{x}, \mathbf{y}))) & \mathbf{A}(\mathbf{x}, \mathbf{y}) \\ \mathbf{A}(\mathbf{x}, \mathbf{y})^T & \operatorname{diag}(\mathbf{c}(\mathbf{A}(\mathbf{x}, \mathbf{y})))\end{bmatrix}, \\ \nabla^2(\tilde{f} - f)(\mathbf{x}, \mathbf{y}) &= \frac{\varepsilon^2}{ne^B}\begin{bmatrix}\operatorname{diag}(e^{\mathbf{x}} + e^{-\mathbf{x}}) & \mathbf{0} \\ \mathbf{0} & \operatorname{diag}(e^{\mathbf{y}} + e^{-\mathbf{y}})\end{bmatrix}.\end{aligned} \tag{3.1}$$

Note that computing $\nabla^2 \tilde{f}(\mathbf{x}, \mathbf{y})$ up to high precision can be done using $\widetilde{O}(m)$ classical queries to $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{y}$. Below we show how to obtain a sparse approximation of $\nabla^2 \tilde{f}(\mathbf{x}, \mathbf{y})$ using only $\widetilde{O}(\sqrt{mn})$ quantum queries. We will do so in the sense of condition (2) of Theorem 3.5 where we take $\mathbf{H}'_m$ to be a (high-precision) additive approximation of $\nabla^2 f(\mathbf{x}, \mathbf{y})$, and $\mathbf{H}'_a = \nabla^2 \tilde{f}(\mathbf{x}, \mathbf{y}) - \mathbf{H}'_m$.

We first obtain a multiplicative spectral approximation of (a high-precision additive approximation of) $\nabla^2 f(\mathbf{x}, \mathbf{y})$. In order to do so we use its structure: it is similar to a Laplacian matrix. This allows us to use the recent quantum Laplacian sparsifier of Apers and de Wolf [6]. For a full proof, carefully keeping track of the bit-complexity, we refer to the full version [17].

▶ **Lemma 3.7.** *Given quantum query access to $\mathbf{x}, \mathbf{y}$ and sparse quantum query access to $\mathbf{A}$, such that $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1 \leq C$, we can compute an SDD matrix $\mathbf{H}_m$ with $\widetilde{O}(n)$ non-zero entries, each off-diagonal entry non-negative, such that there exist symmetric $\mathbf{H}'_m$ and $\mathbf{H}'_{a,f}$ satisfying $\mathbf{H}'_m + \mathbf{H}'_{a,f} = \nabla^2 f(\mathbf{x}, \mathbf{y})$, and $0.9\mathbf{H}_m \preceq \mathbf{H}'_m \preceq 1.1\mathbf{H}_m$, $\|\mathbf{H}'_{a,f}\|_1 \leq \delta_a$, in time $\widetilde{O}(\sqrt{mn}\operatorname{polylog}(C/\delta_a))$.*

Similarly, we can efficiently compute an additive approximation of the Hessian of the regularization term $\tilde{f} - f$ as long as $\mathbf{x}$ and $\mathbf{y}$ have ℓ_∞-norm not much larger than B, using the expression given in Equation (3.1).

▶ **Lemma 3.8.** *Given quantum query access to* $\mathbf{x}, \mathbf{y}$ *with* $\|\mathbf{x}\|_\infty, \|\mathbf{y}\|_\infty \leq B + \ln(4n + (n\ln(1/\mu)/\varepsilon^2))$*, we can compute a non-negative diagonal matrix* $\mathbf{H}_{a,\tilde{f}}$ *that satisfies* $\|\mathbf{H}_{a,\tilde{f}} - \nabla^2(\tilde{f} - f)(\mathbf{x}, \mathbf{y})\|_1 \leq \delta_a$*, in time* $\widetilde{O}(n \log(1/\delta_a\mu) \operatorname{polylog}(\varepsilon))$*.*

▶ **Theorem 3.9.** *Given quantum query access to* $\mathbf{x}, \mathbf{y}$ *with* $\|\mathbf{x}\|_\infty, \|\mathbf{y}\|_\infty \leq B + \ln(4n + (n\ln(1/\mu)/\varepsilon^2))$*, and sparse quantum query access to* $\mathbf{A}$*, if* $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1 \leq C$*, then we can compute (classical descriptions of) an SDD matrix* $\mathbf{H}_m$ *with* $\widetilde{O}(n)$ *non-zero entries, with all of the off-diagonal entries non-negative, and a non-negative diagonal matrix* $\mathbf{H}_a$ *such that there exist symmetric* $\mathbf{H}'_m$*,* $\mathbf{H}'_a$ *with* $\mathbf{H}'_m + \mathbf{H}'_a = \nabla^2 \tilde{f}(\mathbf{x}, \mathbf{y})$ *and*

$$0.9\mathbf{H}_m \preceq \mathbf{H}'_m \preceq 1.1\mathbf{H}_m, \quad \|\mathbf{H}_a - \mathbf{H}'_a\|_1 \leq \delta_a$$

in quantum time $\widetilde{O}(\sqrt{mn} \operatorname{polylog}(C/\mu\delta_a))$*.*

Proof. Let $\mathbf{H}_m$ be the matrix obtained from Lemma 3.7, and let $\mathbf{H}_a$ be the matrix $\mathbf{H}_{a,\tilde{f}}$ obtained from Lemma 3.8. Then $\mathbf{H}$ satisfies the desired properties, with $\mathbf{H}'_m$ as in Lemma 3.7, and $\mathbf{H}'_a = \mathbf{H}'_{a,f} + \nabla^2(\tilde{f} - f)(\mathbf{x}, \mathbf{y})$ with $\mathbf{H}'_{a,f}$ as in Lemma 3.7. ◀

In order to obtain a good approximation of the gradient of $\tilde{f}$, which is given by

$$\nabla \tilde{f}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \mathbf{r}(\mathbf{A}(\mathbf{x}, \mathbf{y})) - \mathbf{r} \\ \mathbf{c}(\mathbf{A}(\mathbf{x}, \mathbf{y})) - \mathbf{c} \end{bmatrix} + \frac{\varepsilon^2}{ne^B} \begin{bmatrix} e^{\mathbf{x}} - e^{-\mathbf{x}} \\ e^{\mathbf{y}} - e^{-\mathbf{y}} \end{bmatrix},$$

we can use similar techniques as the prior work on quantum algorithms for matrix scaling [5]. For computing the i-th row marginal, these are based on a careful implementation of amplitude estimation on the unitary that prepares states that are approximately of the form

$$\sum_j |0\rangle \sqrt{A_{ij} e^{x_i + y_j}}\, |j\rangle + |1\rangle \sqrt{1 - A_{ij} e^{x_i + y_j}}\, |j\rangle\,,$$

assuming that the i-th row of $\mathbf{A}(\mathbf{x}, \mathbf{y})$ is properly normalized. The output is an estimate of the i-th row marginal with multiplicative error $1 \pm \delta$, which translates into additive error $\delta \cdot r_i(\mathbf{A}(\mathbf{x}, \mathbf{y}))$; we refer to [5, Thm. 4.5 (arXiv)] for a more precise statement. The part of the gradient coming from the regularization term is dealt with similarly as in Lemma 3.8.

▶ **Lemma 3.10.** *Given quantum query access to* $\mathbf{x}, \mathbf{y}$ *and sparse quantum query access to* $\mathbf{A}$*, if* $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1 \leq C$*, we can find a classical description of a vector* $\mathbf{b} \in \mathbb{R}^n$ *such that* $\|\mathbf{b} - \nabla \tilde{f}(\mathbf{x}, \mathbf{y})\|_1 \leq \delta \cdot C$ *in quantum time* $\widetilde{O}(\sqrt{mn}/\delta \cdot \operatorname{polylog}(C/\mu))$*.*

The following lemma and corollary help us ensure that throughout the algorithm, $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1$ is bounded above by a constant; if $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1$ is too large, we can change the overall scaling of the matrix and decrease the regularized potential (so in particular, we stay in the sublevel set of the regularized potential).

▶ **Lemma 3.11.** *Let* $\mathbf{x}, \mathbf{y}$ *be such that* $\tilde{f}(\mathbf{x}, \mathbf{y}) \leq \tilde{f}(\mathbf{0}, \mathbf{0})$*, and assume* $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1 \geq C'$ *where* $C' > 1$*. Let* $\mathbf{x}' = \mathbf{x} - \ln(\gamma)\mathbf{1}$ *where* $1 \leq \gamma \leq C'$*. Then* $\tilde{f}(\mathbf{x}', \mathbf{y}) - \tilde{f}(\mathbf{x}, \mathbf{y}) \leq (\frac{1}{\gamma} - 1)C' + \ln(\gamma) + (\gamma - 1)\left(\ln(1/\mu) + \frac{4\varepsilon^2}{e^B}\right)$*.*

Proof. We have

$$\tilde{f}(\mathbf{x}', \mathbf{y}) - \tilde{f}(\mathbf{x}, \mathbf{y})$$
$$= \left(\frac{1}{\gamma} - 1\right) \|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1 + \ln(\gamma) + \frac{\varepsilon^2}{ne^B} \left(\frac{1}{\gamma} - 1\right) \left(\sum_i e^{x_i}\right) + \frac{\varepsilon^2}{ne^B} (\gamma - 1)(\sum_i e^{-x_i})$$

$$\leq \left(\frac{1}{\gamma} - 1\right) \|\mathbf{A}(\mathbf{x},\mathbf{y})\|_1 + \ln(\gamma) + 0 + \frac{\varepsilon^2}{ne^B}(\gamma - 1)(\sum_i e^{-x_i})$$
$$\leq \left(\frac{1}{\gamma} - 1\right) C' + \ln(\gamma) + (\gamma - 1)\left(\ln(1/\mu) + \frac{4\varepsilon^2}{e^B}\right)$$

where for the last inequality we use $\|\mathbf{A}(\mathbf{x},\mathbf{y})\|_1 \geq C'$ for the first term and Equation (B.5) for the last term. ◀

An appropriate choice of C' and γ makes the bound in the above lemma non-positive.

▶ **Corollary 3.12.** *Let $\varepsilon \leq 1$ and $\mu \leq 1$, set $\gamma = 2$ and $C' = 2(\ln(2/\mu) + 4\varepsilon^2/e^B)$. Then, if $\|\mathbf{A}(\mathbf{x},\mathbf{y})\|_1 \geq C'$ and $\tilde{f}(\mathbf{x},\mathbf{y}) \leq \tilde{f}(\mathbf{0},\mathbf{0})$, we have $\tilde{f}(\mathbf{x}',\mathbf{y}) \leq \tilde{f}(\mathbf{x},\mathbf{y})$.*

3.3 Quantum box-constrained scaling

Combining the above leads to a quantum algorithm for matrix scaling that is based on classical box-constrained Newton methods. See Algorithm 1 for its formal definition. In Theorem 3.13 we analyze its output.

Algorithm 1 Quantum box-constrained Newton method for matrix scaling.

```
Input: Oracle access to $\mathbf{A} \in [\mu, 1]^{n\times n}$ with $\|\mathbf{A}\|_1 \leq 1$ and $\mu > 0$, error $\varepsilon > 0$, targets
       $\mathbf{r}, \mathbf{c} \in \mathbb{R}^n_{>0}$ with $\|\mathbf{r}\|_1 = 1 = \|\mathbf{c}\|_1$, diameter bound $B \geq 1$, classical $k$-oracle $\mathcal{A}$
       for SDD matrices with non-negative off-diagonal entries
Output: Vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with $\|(\mathbf{x},\mathbf{y})\|_\infty \leq B + \ln(4n + (n\ln(1/\mu)/\varepsilon^2))$
set $T = \lceil 4e^4 \max(kB + \ln(4n + (n\ln(1/\mu)/\varepsilon^2)), 1) \cdot \ln\left(\frac{\ln(1/\mu)+2\varepsilon^2/e^B}{\varepsilon^2/2}\right)\rceil$;
set $C' = 2\lceil \ln(2/\mu) + 8\varepsilon^2/e^B \rceil$;
set $\varepsilon' = \lfloor \varepsilon^2/8e^4 \max(k(B + \ln(4n + (n\ln(1/\mu)/\varepsilon^2))), 1) \rfloor$;
store $\mathbf{x}^{(0)}, \mathbf{y}^{(0)} = \mathbf{0} \in \mathbb{R}^n$ in QCRAM;
for $i = 0, \ldots, T-1$ do
    compute $\mathbf{H}_m$, $\mathbf{H}_a$ s.t. $\mathbf{H}_m + \mathbf{H}_a \approx \nabla^2 \tilde{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ as in Theorem 3.9 with
      $\delta_a = \varepsilon' k^2/2e^2$;
    compute $\mathbf{b} \approx \nabla \tilde{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ as in Lemma 3.10 at $\mathbf{x}^{(i)}, \mathbf{y}^{(i)}$ with $\delta = \varepsilon'/3$;
    compute $\Delta = \mathcal{A}(\frac{4e^2}{3k^2} \cdot (\mathbf{H}_m + \mathbf{H}_a), \frac{\mathbf{b}}{k})$;
    compute $(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)}) = (\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) + \frac{1}{k}\Delta$ and store in QCRAM;
    set flag = true;
    while flag do
        Compute $C'/2$-additive approximation $\gamma$ of $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1$;
        if $\gamma \leq 3C'/2$ then
            set flag = false;
        else
            update $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i+1)} - \ln(2)\mathbf{1}$ in QCRAM;
        end if
    end while
end for
return $(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{(T)}, \mathbf{y}^{(T)})$;
```

▶ **Theorem 3.13.** *Let $\mathbf{A} \in [0,1]^{n\times n}$ with m non-zero entries, $\mathbf{r}, \mathbf{c} \in \mathbb{R}^n_{>0}$ such that $\|\mathbf{r}\|_1 = 1 = \|\mathbf{c}\|$, and assume $\mathbf{A}$ is asymptotically $(\mathbf{r}, \mathbf{c})$-scalable. Let $\varepsilon > 0$, let $B \geq 1$, and assume there exist $(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon)$ such that $\|(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon)\|_\infty \leq B$ and $f(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon) - f^* \leq \varepsilon^2$. Furthermore, let $\mathcal{A}$ be the $O(\log(n))$-oracle of Theorem 3.4. Then Algorithm 1 with these parameters outputs, with probability $\geq 2/3$, vectors $\mathbf{x}, \mathbf{y}$ such that $f(\mathbf{x}, \mathbf{y}) - f^* \leq 6\varepsilon^2$ and runs in quantum time $\widetilde{O}\big(B^2\sqrt{mn}/\varepsilon^2\big)$.*

Proof. In every iteration, the matrices $\mathbf{H}_m, \mathbf{H}_a$ and the vector $\mathbf{b}$ are such that they satisfy the requirements of Theorem 3.5, hence

$$\tilde{f}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)}) - \tilde{f}^* \leq \left(1 - \frac{1}{4e^4 \max(kR_\infty, 1)}\right)(\tilde{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \tilde{f}^*) + \frac{e^2\delta_a}{k^2} + \frac{3\delta}{2}$$

where $R_\infty \leq B + \ln(4n + (n\ln(1/\mu)/\varepsilon^2)))$ is the ℓ_∞-radius of the sublevel set $\{(\mathbf{x}, \mathbf{y}) : \tilde{f}(\mathbf{x}, \mathbf{y}) \leq \tilde{f}(\mathbf{0}, \mathbf{0})\}$ about $(\mathbf{0}, \mathbf{0})$, whose upper bound follows from Lemma 3.6. From here on, we write $M = 4e^4 \max(kR_\infty, 1)$. The choice of δ_a and δ in the algorithm is such that $e^2\delta_a/k^2 + 3\delta/2 \leq \frac{\varepsilon^2}{2M}$, hence we can also bound the progress by

$$\tilde{f}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)}) - \tilde{f}^* \leq \left(1 - \frac{1}{M}\right)(\tilde{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \tilde{f}^*) + \frac{\varepsilon^2}{2M}.$$

Corollary 3.12 shows that if $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1$ is larger than C', then we can shift $\mathbf{x}$ by $-\ln(2)\mathbf{1}$, this halves $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1$ and does not increase the regularized potential. Repeating this roughly $\log_2(\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1/C')$ many times[2] reduces $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1$ to at most $C = 2C'$. Determining when to stop this process requires a procedure to distinguish between the cases $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1 \leq C'$ and $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1 \geq 2C'$ (if in between C' and $2C'$ either continuing or stopping is fine). Such a procedure can be implemented by computing a $C'/2$-additive approximation of $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1$, which can be done using $\widetilde{O}(\sqrt{mn}\,\mathrm{polylog}(C'/\mu))$ quantum queries, see (the proof of) [5, Lemma 4.6 (arXiv)]. Therefore, throughout the algorithm we may assume that $\|\mathbf{A}(\mathbf{x}^{(i+1)}, \mathbf{y}^{(i+1)})\|_1 \leq 2C' = C$.

It remains to show that $\tilde{f}(\mathbf{x}^{(T)}, \mathbf{y}^{(T)}) - \tilde{f}^* \leq \varepsilon^2$ for our choice of T. Note that we have

$$\begin{aligned}
\tilde{f}(\mathbf{x}^{(T)}, \mathbf{y}^{(T)}) - \tilde{f}^* &\leq \left(1 - \frac{1}{M}\right)^T (\tilde{f}(\mathbf{0}, \mathbf{0}) - \tilde{f}^*) + \sum_{i=0}^{T-1} \left(1 - \frac{1}{M}\right)^{T-i-1} \cdot \frac{\varepsilon^2}{2M} \\
&\leq \left(1 - \frac{1}{M}\right)^T (\tilde{f}(\mathbf{0}, \mathbf{0}) - \tilde{f}^*) + \left(1 - (1 - \frac{1}{M})^T\right) \cdot \frac{\varepsilon^2}{2} \\
&\leq \left(1 - \frac{1}{M}\right)^T \left(f(\mathbf{0}, \mathbf{0}) - f^* + \frac{2\varepsilon^2}{e^B}\right) + \frac{\varepsilon^2}{2} \\
&\leq \left(1 - \frac{1}{M}\right)^T \left(\ln(1/\mu) + \frac{2\varepsilon^2}{e^B}\right) + \frac{\varepsilon^2}{2} \leq \varepsilon^2
\end{aligned}$$

where in the third inequality we use Lemma 3.6, and in the last inequality we use

$$\begin{aligned}
T &= \left\lceil 4e^4 \max(kB + \ln(4n + (n\ln(1/\mu)/\varepsilon^2)), 1) \cdot \ln\left(\frac{\ln(1/\mu) + 2\varepsilon^2/e^B}{\varepsilon^2/2}\right)\right\rceil \\
&\geq \left\lceil M \cdot \ln\left(\frac{\ln(1/\mu) + 2\varepsilon^2/e^B}{\varepsilon^2/2}\right)\right\rceil \geq \frac{1}{\ln(1 - \frac{1}{M})} \cdot \ln\left(\frac{\varepsilon^2/2}{\ln(1/\mu) + \frac{2\varepsilon^2}{e^B}}\right).
\end{aligned}$$

[2] Which is an almost constant number of times: in a single update of the box-constrained method, we take steps of size at most 1 in ℓ_∞-norm, so individual entries can only grow by a factor e^2 in a single iteration, and the holds same for $\|\mathbf{A}(\mathbf{x}, \mathbf{y})\|_1$.

This implies that $f(\mathbf{x}^{(T)}, \mathbf{y}^{(T)}) - f^* \leq \tilde{f}(\mathbf{x}^{(T)}, \mathbf{y}^{(T)}) - \tilde{f}^* + 5\varepsilon^2 \leq 6\varepsilon^2$, where we crucially use the last point of Lemma 3.6 and the assumption that there exist $(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon)$ with $\|(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon)\|_\infty \leq B$ which ε^2-minimize f.

Finally we bound the time complexity of Algorithm 1. For each of the quoted results, we use the choice $C = 2C' = \widetilde{O}\big(\ln(n) + \varepsilon^2\big)$. In each of the T iterations we compute:

1. approximations $\mathbf{H}_m$, $\mathbf{H}_a$ of $\nabla^2 \tilde{f}(\mathbf{x}^{(i)}, \mathbf{x}^{(i)})$ in time $\widetilde{O}(\sqrt{mn}\,\mathrm{polylog}(1/\varepsilon))$ (using that C, $1/\mu$ are at most $\mathrm{poly}(n)$),
2. an $\varepsilon'/3$-ℓ_1-approximation of $\nabla\tilde{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ in time $\widetilde{O}(\sqrt{mn}/\varepsilon') = \widetilde{O}\big(B\sqrt{mn}/\varepsilon^2\big)$,
3. an update Δ in time $\widetilde{O}(n)$ using one call to the $k = O(\log(n))$-oracle on SDD-matrices with $\widetilde{O}(n)$ non-zero entries from Theorem 3.4,
4. at most $O(1)$ many times (using the fact that in Algorithm 1 the 1-norm changes by at most a constant factor since $\|\frac{1}{k}\Delta\|_\infty \leq 1$) an $O\big(\ln(1/\mu) + \varepsilon^2\big)$-additive approximation of $\|\mathbf{A}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\|_1$ in time $\widetilde{O}(\sqrt{mn})$.

Note that the second contribution dominates the others, resulting in an overall time complexity of $\widetilde{O}\big(B^2\sqrt{mn}/\varepsilon^2\big)$. ◀

The above proof relies on Theorem 3.5 to show that the (regularized) potential decreases in each iteration. This decrease depends on the precision used for the marginal estimation in that iteration and one can show that the choice of precision in Algorithm 1 is asymptotically optimal, see the full version [17].

Algorithm 1 takes as (part of the) input a bound B on the ℓ_∞-norm of an ε^2-minimizer of f. For the purpose of matrix scaling, one can avoid knowing such a bound in advance, by running the algorithm for successive powers of 2 (i.e., $B = 1$, $B = 2$, $B = 4,\ldots$) and testing whether the output yields an ε-scaling or not. Verifying whether given $\mathbf{x}, \mathbf{y}$ yield an ε-scaling of $\mathbf{A}$ can be done in time $\widetilde{O}\big(\sqrt{mn}/\varepsilon^2\big)$. Note that this gives an algorithm for ε-scaling whose complexity depends on a diameter bound for ε^2-minimizers of f, rather than a diameter bound for ε-scaling vectors. Furthermore, such an approach does not work for the task of finding an ε^2-minimizer of f, as we do not know how to test this property efficiently.

▶ **Corollary 3.14.** *For asymptotically-scalable matrices $\mathbf{A} \in \mathbb{R}^{n\times n}_{\geq 0}$ with m non-zero entries, one can find $O(\varepsilon)$-ℓ_1-scaling vectors $(\mathbf{x}, \mathbf{y})$ of $\mathbf{A}$ to target marginals $\mathbf{r}, \mathbf{c} \in \mathbb{R}^n_{>0}$ with $\|\mathbf{r}\|_1 = 1 = \|\mathbf{c}\|_1$ in time $\widetilde{O}\big(R_\infty^2\sqrt{mn}/\varepsilon^2\big)$, where R_∞ is such that there exists an ε^2-approximate minimizer $(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon)$ of f with $R_\infty = \|(\mathbf{x}_\varepsilon, \mathbf{y}_\varepsilon)\|_\infty + \ln(4n + (n\ln(1/\mu)/\varepsilon^2))$.*

For the general case mentioned above, we do not have good (i.e., polylogarithmic) bounds on the parameter R_∞. We do have such bounds when $\mathbf{A}$ is entrywise positive: it is well-known that such an $\mathbf{A}$ can be exactly scaled to uniform marginals with scaling vectors $(\mathbf{x}, \mathbf{y})$ such that $\|(\mathbf{x}, \mathbf{y})\|_\infty = O(\log(\|\mathbf{A}\|_1/\mu))$ (cf. [22, Lem. 1], [12, Lem. 4.11]). In particular, this implies that there exists a minimizer $(\mathbf{x}^*, \mathbf{y}^*)$ of f with $\|(\mathbf{x}^*, \mathbf{y}^*)\|_\infty = O(\log(\|\mathbf{A}\|_1/\mu)) = \widetilde{O}(1)$ and therefore we have the following corollary.

▶ **Corollary 3.15.** *For entrywise-positive matrices $\mathbf{A}$, one can find an ε-ℓ_1-scaling of $\mathbf{A}$ to uniform marginals in time $\widetilde{O}\big(n^{1.5}/\varepsilon^2\big)$.*

References

1 Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 890–901, 2017. arXiv:1704.02315.

2 Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in Neural Information Processing Systems*, volume 30, pages 1964–1974, 2017. arXiv:1705.09634.

3 Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. arXiv:quant-ph/0002066.

4 Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.

5 Joran van Apeldoorn, Sander Gribling, Yinan Li, Harold Nieuwboer, Michael Walter, and Ronald de Wolf. Quantum Algorithms for Matrix Scaling and Matrix Balancing. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198, pages 110:1–110:17, 2021. arXiv:2011.12823v1, doi:10.4230/LIPIcs.ICALP.2021.110.

6 Simon Apers and Ronald de Wolf. Quantum speedup for graph sparsification, cut approximation and laplacian solving. *Proceedings of IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS'20)*, pages 637–648, 2020. arXiv:1911.07306.

7 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001. arXiv:quant-ph/9802049.

8 Yvonne M. M. Bishop, Stephen E. Fienberg, and Paul W. Holland. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, 1975.

9 David T. Brown. A note on approximations to discrete probability distributions. *Information and control*, 2(4):386–392, 1959.

10 Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Oliveira, Michael Walter, and Avi Wigderson. Towards a theory of non-commutative optimization: geodesic 1st and 2nd order methods for moment maps and polytopes. In *Proceedings of 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS'19)*, pages 845–861, 2019. arXiv:1910.12375.

11 Deeparnab Chakrabarty and Sanjeev Khanna. Better and simpler error analysis of the Sinkhorn–Knopp algorithm for matrix scaling. *Mathematical Programming*, pages 1–13, 2020.

12 Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained Newton's method and interior point methods. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 902–913, 2017. doi:10.1109/FOCS.2017.88.

13 Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pages 2292–2300, 2013. arXiv:1306.0895.

14 W. Edwards Deming and Frederick F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11(4):427–444, 1940.

15 Pavel E. Dvurechensky, Alexander V. Gasnikov, and Alexey Kroshnin. Computational Optimal Transport: Complexity by Accelerated Gradient Descent Is Better Than by Sinkhorn's Algorithm. In *ICML*, 2018. arXiv:1802.04367.

16 Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. Operator scaling: theory and applications. *Foundations of Computational Mathematics*, pages 1–68, 2019. Earlier version in FOCS'16.

17 Sander Gribling and Harold Nieuwboer. Improved quantum lower and upper bounds for matrix scaling, 2021. arXiv:2109.15282v1.

18 Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Annual ACM SIGACT Symposium on Theory of Computing (STOC'07)*, pages 526–535, 2007. arXiv:quant-ph/0611054, doi:10.1145/1250790.1250867.

19 Martin Idel. A review of matrix scaling and Sinkhorn's normal form for matrices and positive maps, 2016. arXiv:1609.06349.

20 B. Kalantari, I. Lari, F. Ricca, and B. Simeone. On the complexity of general matrix scaling and entropy minimization via the RAS algorithm. *Mathematical Programming*, 112:371–401, 2008.

21 Bahman Kalantari and Leonid Khachiyan. On the rate of convergence of deterministic and randomized RAS matrix scaling algorithms. *Operations Research Letters*, 14(5):237–244, 1993. doi:10.1016/0167-6377(93)90087-W.

22 Bahman Kalantari and Leonid Khachiyan. On the complexity of nonnegative-matrix scaling. *Linear Algebra and its Applications*, 240:87–103, 1996. doi:10.1016/0024-3795(94)00188-X.

23 J. Kruithof. Telefoonverkeersrekening. *De Ingenieur*, 52:E15–E25, 1937.

24 Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified Cholesky and multigrid solvers for connection Laplacians. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC'16)*, pages 842–850, 2016. arXiv:1512.01892, doi:10.1145/2897518.2897640.

25 Troy Lee, Rajat Mittal, Ben Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. *Proceedings of IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*, pages 344–353, 2011. arXiv:1011.3020, doi:10.1109/FOCS.2011.75.

26 Troy Lee and Jérémie Roland. A strong direct product theorem for quantum query complexity. *Computational Complexity*, 22(2):429–462, 2013. Earlier version in CCC'12. arXiv:1104.4468.

27 Yin Tat Lee, Richard Peng, and Daniel A. Spielman. Sparsified Cholesky solvers for SDD linear systems, 2015. arXiv:1506.08204.

28 Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000.

29 Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st Annual ACM SIGACT Symposium on Theory of Computing (STOC'99)*, pages 384–393, 1999. arXiv:quant-ph/9804066, doi:10.1145/301250.301349.

30 Uriel G. Rothblum and Hans Schneider. Scalings of matrices which have prespecified row sums and column sums via optimization. *Linear Algebra and its Applications*, 114:737–764, 1989.

31 Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

A Missing proofs for Section 2

A.1 Strong convexity properties of the potential

Lemma 2.8 shows that a vector $(\mathbf{x},\mathbf{y}) \in V$ for which $\|\nabla f(\mathbf{x},\mathbf{y})\|_2$ is small, is close to the minimizer of f. Here we prove this lemma (see Corollary A.6) using strong convexity properties of f. In Lemma A.1 we show that the Hessian of f restricted to V has smallest eigenvalue at least $n \cdot \mu(\mathbf{x},\mathbf{y})$ where $\mu(\mathbf{x},\mathbf{y})$ is the smallest entry appearing in $(A_{ij}e^{x_i+y_j})_{i,j}$. In Lemma A.3 we show that $\mu(\mathbf{x}^*,\mathbf{y}^*) = \Theta(1/n^2)$. This implies that $\mu(\mathbf{x},\mathbf{y}) = \Theta(1/n^2)$ for all $(\mathbf{x},\mathbf{y})$ that are a constant distance away from $(\mathbf{x}^*,\mathbf{y}^*)$ in the ℓ_∞-norm, in other words, f is $\Theta(1/n)$-strongly convex around its minimizer. Lemma A.5 summarizes these lemmas: it gives a quantitative bound on the distance to a minimizer, in terms of the gradient.

▶ **Lemma A.1.** *Let $\mathbf{A}$ be an entrywise non-negative $n \times n$ matrix and let $f\colon V \subset \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the potential for this matrix as given in* (1.2)*, where V is the orthogonal complement of $(\mathbf{1}_n, -\mathbf{1}_n)$. Then $\nabla^2 f(\mathbf{x},\mathbf{y}) \succeq \mu(\mathbf{x},\mathbf{y}) \cdot n \cdot \mathbf{P}_V$ where $\mathbf{P}_V$ is the projection onto V and $\mu(\mathbf{x},\mathbf{y})$ is the smallest entry appearing in $\mathbf{A}(\mathbf{x},\mathbf{y})$. In particular, f is strictly convex on V.*

Proof. The Hessian of the potential $f(\mathbf{x},\mathbf{y}) = \sum_{i,j=1}^n A_{ij}e^{x_i+y_j} - \langle \mathbf{r},\mathbf{x}\rangle - \langle \mathbf{c},\mathbf{y}\rangle$ is given by

$$\nabla^2 f(\mathbf{x},\mathbf{y}) = \begin{bmatrix} \operatorname{diag}(\mathbf{r}(\mathbf{A}(\mathbf{x},\mathbf{y})) & \mathbf{A}(\mathbf{x},\mathbf{y}) \\ \mathbf{A}(\mathbf{x},\mathbf{y})^T & \operatorname{diag}(\mathbf{c}(\mathbf{A}(\mathbf{x},\mathbf{y}))) \end{bmatrix}.$$

We give a lower bound on the non-zero eigenvalues of the Hessian as follows. Conjugating the Hessian with the $2n \times 2n$ matrix $\operatorname{diag}(\mathbf{I}, -\mathbf{I})$ preserves the spectrum, i.e., changing the signs of the off-diagonal $\mathbf{A}(\mathbf{x}, \mathbf{y})$ blocks yields a matrix which one can recognize as the weighted Laplacian of a complete bipartite graph. We denote by $\mu(\mathbf{x}, \mathbf{y})$ the smallest entry of $\mathbf{A}(\mathbf{x}, \mathbf{y})$ and we use $\mathbf{J}$ for the $n \times n$ all-ones matrix. Then

$$\begin{bmatrix} \operatorname{diag}(\mathbf{r}(\mathbf{A}(\mathbf{x}, \mathbf{y})) & -\mathbf{A}(\mathbf{x}, \mathbf{y}) \\ -\mathbf{A}(\mathbf{x}, \mathbf{y})^T & \operatorname{diag}(\mathbf{c}(\mathbf{A}(\mathbf{x}, \mathbf{y}))) \end{bmatrix} \succeq \begin{bmatrix} n\mu(\mathbf{x}, \mathbf{y})\mathbf{I} & -\mu(\mathbf{x}, \mathbf{y})\mathbf{J} \\ -\mu(\mathbf{x}, \mathbf{y})\mathbf{J} & n\mu(\mathbf{x}, \mathbf{y})\mathbf{I} \end{bmatrix} = \mu(\mathbf{x}, \mathbf{y}) \begin{bmatrix} n\mathbf{I} & -\mathbf{J} \\ -\mathbf{J} & n\mathbf{I} \end{bmatrix},$$

where the PSD inequality follows because the difference of the terms is the weighted Laplacian of the bipartite graph with weighted bipartite adjacency matrix $\mathbf{A}(\mathbf{x}, \mathbf{y}) - \mu(\mathbf{x}, \mathbf{y})\mathbf{J}$, which has non-negative entries. Now observe that the last term $\begin{bmatrix} n\mathbf{I} & -\mathbf{J} \\ -\mathbf{J} & n\mathbf{I} \end{bmatrix}$ is the (unweighted) Laplacian of the complete bipartite graph $K_{n,n}$, whose spectrum is $2n$, n, 0 with multiplicities 1, $2n-2$ and 1 respectively. The zero eigenvalue corresponds to the all-ones vector of length $2n$ and it is easy to see that indeed $(\mathbf{1}, -\mathbf{1})$ also lies in the kernel of $\nabla^2 f(\mathbf{x}, \mathbf{y})$. This shows that the non-zero eigenvalues of $\nabla^2 f(\mathbf{x}, \mathbf{y})$ are at least $n \cdot \mu(\mathbf{x}, \mathbf{y})$, and that it has a one-dimensional eigenspace corresponding to 0, spanned by the vector $(\mathbf{1}, -\mathbf{1})$. Hence, $\nabla^2 f(\mathbf{x}, \mathbf{y}) \succeq \mu(\mathbf{x}, \mathbf{y}) \cdot n \cdot \mathbf{P}_V$. ◀

We now bound the smallest entry of the rescaled matrix. For this we use the following lemma (cf. [20, Lem. 6.2], [5, Cor. C.3 (arXiv)]) which bounds the variation norm of the scaling vectors $(\mathbf{x}^*, \mathbf{y}^*)$ of an exact scaling.

▶ **Lemma A.2.** *Let $\mathbf{A} \in [\mu, \nu]^{n \times n}$ and let $(\mathbf{x}^*, \mathbf{y}^*) \in \mathbb{R}^n \times \mathbb{R}^n$ be such that $\mathbf{A}(\mathbf{x}^*, \mathbf{y}^*)$ is exactly $(\mathbf{r}, \mathbf{c})$-scaled. Then*

$$x^*_{\max} - x^*_{\min} \leq \ln \frac{\nu}{\mu} + \ln \frac{r_{\max}}{r_{\min}} \quad \text{and} \quad y^*_{\max} - y^*_{\min} \leq \ln \frac{\nu}{\mu} + \ln \frac{c_{\max}}{c_{\min}}.$$

▶ **Lemma A.3.** *Let $\mathbf{A} \in [\mu, \nu]^{n \times n}$ be an entrywise-positive matrix with $\|\mathbf{A}\|_1 = 1$ and let $f \colon V \subset \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the potential for this matrix as given in (1.2), where V is the orthogonal complement of $(\mathbf{1}_n, -\mathbf{1}_n)$. Let $(\mathbf{x}^*, \mathbf{y}^*) \in V$ be the unique minimizer of f in V. Then $\mu(\mathbf{x}^*, \mathbf{y}^*) \geq \frac{1}{n^2} \left(\frac{\mu}{\nu}\right)^3$. Moreover, for every $(\mathbf{x}, \mathbf{y}) \in V$ we have $\mu(\mathbf{x}, \mathbf{y}) \geq \mu(\mathbf{x}^*, \mathbf{y}^*) e^{-2\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty}$.*

Proof. By Lemma A.1 f is strictly convex on V. We also know that $\mathbf{A}$ is exactly scalable. Hence f has a unique minimizer $(\mathbf{x}^*, \mathbf{y}^*)$. By Lemma A.2 we know that the variation norm of x^* and y^* are bounded by $\ln(\nu/\mu)$. Hence, for every $i, i', j, j' \in [n]$ we have

$$\left| \ln \left(\frac{e^{x^*_i + y^*_j}}{e^{x^*_{i'} + y^*_{j'}}} \right) \right| \leq |x^*_i - x^*_{i'}| + |y^*_j - y^*_{j'}| = 2 \ln(\nu/\mu).$$

Therefore, the ratio between entries of $\mathbf{A}(\mathbf{x}^*, \mathbf{y}^*)$ is bounded:

$$\left| \frac{\mathbf{A}(\mathbf{x}^*, \mathbf{y}^*)_{ij}}{\mathbf{A}(\mathbf{x}^*, \mathbf{y}^*)_{i'j'}} \right| \leq \left| \frac{A_{ij}}{A_{i'j'}} \right| \left| \left(\frac{e^{x^*_i + y^*_j}}{e^{x^*_{i'} + y^*_{j'}}} \right) \right| \leq \frac{\nu}{\mu} e^{2 \ln(\nu/\mu)} = \left(\frac{\nu}{\mu} \right)^3.$$

Since the sum of the entries of $\mathbf{A}(\mathbf{x}^*, \mathbf{y}^*)$ equals 1, this implies that the smallest entry of $\mathbf{A}(\mathbf{x}^*, \mathbf{y}^*)$ is at least $\mu(\mathbf{x}^*, \mathbf{y}^*) \geq \frac{1}{n^2} \left(\frac{\mu}{\nu}\right)^3$. Finally, for $(\mathbf{x}, \mathbf{y}) \in V$ and all $i, j \in [n]$ we have $A_{ij} e^{x_i + y_j} \geq A_{ij} e^{x^*_i + y^*_j - 2\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty}$, so taking the minimum over all i, j gives $\mu(\mathbf{x}, \mathbf{y}) \geq \mu(\mathbf{x}^*, \mathbf{y}^*) e^{-2\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty}$. ◀

Finally, to obtain a diameter bound for the set of points with a small gradient we will use the following (well-known) lemma.

▶ **Lemma A.4.** *Assume $g\colon \mathbb{R}^d \to \mathbb{R}$ is a C^2 convex function such that $\nabla g(\mathbf{0}) = 0$, and assume that for all $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{x}\|_\infty \leq r$, we have $\nabla^2 g(\mathbf{x}) \succeq \lambda I$. Then*

$$\|\nabla g(\mathbf{x})\|_2 \geq \lambda \|\mathbf{x}\|_2 \min(1, r/\|\mathbf{x}\|_\infty) \geq \lambda \min(\|\mathbf{x}\|_\infty, r).$$

In particular, to guarantee that $\|\mathbf{x}\|_\infty \leq C$ for $C \geq 0$, it suffices to show that $\|\nabla g(\mathbf{x})\|_2 < \lambda \min(C, r)$ (strict inequality is necessary as it forces $\min(\|\mathbf{x}\|_\infty, r) = \|\mathbf{x}\|_\infty$).

Proof. Fix $\mathbf{x} \in \mathbb{R}^n$ and consider $h\colon \mathbb{R} \to \mathbb{R}$ defined by $h(t) = g(t\mathbf{x})$. Then h is convex, $\partial_{t=0} h(t) = 0$ and $\partial^2_{t=s} h(t) \geq 0$ for all $s \in \mathbb{R}$. Now assume for $s \in \mathbb{R}$ that $|s|\|\mathbf{x}\|_\infty \leq r$. Then

$$\partial^2_{t=s} h(t) = \partial_{t=s}(Dg(t\mathbf{x})[\mathbf{x}]) = D^2 g(s\mathbf{x})[\mathbf{x},\mathbf{x}] = \mathbf{x}^T \nabla^2 g(s\mathbf{x})\mathbf{x} \geq \lambda \|\mathbf{x}\|_2^2.$$

For $s \geq 0$ this yields a lower bound on $\langle \nabla g(s\mathbf{x}), \mathbf{x}\rangle$ of the form

$$\begin{aligned}\langle \nabla g(s\mathbf{x}), \mathbf{x}\rangle = \partial_{t=s} h(t) = \int_0^s \partial^2_{t=\tau} h(t)\, d\tau &\geq \int_0^{\min(s, r/\|\mathbf{x}\|_\infty)} \partial^2_{t=\tau} h(t)\, d\tau \\ &\geq \int_0^{\min(s, r/\|\mathbf{x}\|_\infty)} \lambda \|\mathbf{x}\|_2^2\, d\tau = \lambda \|\mathbf{x}\|_2^2 \min(s, r/\|\mathbf{x}\|_\infty),\end{aligned}$$

where the first inequality follows from the convexity of h. Setting $s = 1$ and using the Cauchy–Schwarz inequality gives $\|\nabla g(\mathbf{x})\|_2 \|\mathbf{x}\|_2 \geq \lambda \|\mathbf{x}\|_2^2 \min(1, r/\|\mathbf{x}\|_\infty)$ so

$$\|\nabla g(\mathbf{x})\|_2 \geq \lambda \|\mathbf{x}\|_2 \min(1, r/\|\mathbf{x}\|_\infty) \geq \lambda \|\mathbf{x}\|_\infty \min(1, r/\|\mathbf{x}\|_\infty) = \lambda \min(\|\mathbf{x}\|_\infty, r). \quad ◀$$

▶ **Lemma A.5.** *Let $\mathbf{A} \in [\mu, \nu]^{n \times n}$ be an entrywise non-negative matrix with $\|\mathbf{A}\|_1 = 1$ and let $f\colon V \subset \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the potential for this matrix as given in (1.2), where V is the orthogonal complement of $(\mathbf{1}_n, -\mathbf{1}_n)$. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be the unique minimizer of f in V and let $0 < \delta < 1$. Let $(\mathbf{x}, \mathbf{y}) \in V$ be such that $\|\nabla f(\mathbf{x}, \mathbf{y})\|_2 < \delta \cdot \frac{1}{n} \left(\frac{\mu}{\nu}\right)^3 e^{-2}$. Then $\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq \delta$.*

Proof. Lemma A.1 shows that $\nabla^2 f(\mathbf{x}, \mathbf{y}) \succeq n \cdot \mu(\mathbf{x}, \mathbf{y}) \cdot \mathbf{P}_V$, where $\mathbf{P}_V$ is the orthogonal projector on V. Lemma A.3 shows that $\mu(\mathbf{x}, \mathbf{y}) \geq \mu(\mathbf{x}^*, \mathbf{y}^*) e^{-2\|(\mathbf{x},\mathbf{y})-(\mathbf{x}^*,\mathbf{y}^*)\|_\infty} \geq \frac{1}{n^2}\left(\frac{\mu}{\nu}\right)^3 e^{-2\|(\mathbf{x},\mathbf{y})-(\mathbf{x}^*,\mathbf{y}^*)\|_\infty}$. Hence, for $(\mathbf{x}, \mathbf{y})$ with $\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq 1$, we have $\nabla^2 f(\mathbf{x}, \mathbf{y}) \succeq \frac{1}{n}\left(\frac{\mu}{\nu}\right)^3 e^{-2} \cdot \mathbf{P}_V$. It then follows from Lemma A.4 that if $\|\nabla f(\mathbf{x}, \mathbf{y})\|_2 < \delta \cdot \frac{1}{n}\left(\frac{\mu}{\nu}\right)^3 e^{-2}$, then $\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq \delta$. ◀

Observe that for $\mathbf{A}^\sigma$ the ratio between its largest and smallest entry is $\frac{b+1}{b-1} \leq 3$. This gives the following corollary, proving Lemma 2.8.

▶ **Corollary A.6.** *Let $\mathbf{A}^\sigma$ be as in Section 2.1 and let f be the associated potential. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be the unique exact scaling of $\mathbf{A}^\sigma$ in V. If $(\mathbf{x}, \mathbf{y}) \in V$ is such that $\|\nabla f(\mathbf{x}, \mathbf{y})\|_2 < \frac{\delta}{27ne^2}$, then $\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|_\infty \leq \delta$.*

B Missing proofs for Section 3

B.1 Minimizing a second-order robust function

Before giving the proof of Theorem 3.5, we introduce the following notation. For a symmetric matrix $\mathbf{H}$ and $\mathbf{b}, \mathbf{z} \in \mathbb{R}^n$, we denote

$$Q(\mathbf{H}, \mathbf{b}, \mathbf{z}) = \langle \mathbf{b}, \mathbf{z}\rangle + \frac{1}{2}\mathbf{z}^T \mathbf{H}\mathbf{z}.$$

We will use the following easily-verified properties of Q repeatedly.

▶ **Lemma B.1.** *For symmetric matrices $\mathbf{H}, \mathbf{H}'$ and vectors $\mathbf{b}, \mathbf{b}', \mathbf{z}$, we have the following estimates:*

1. *If $\mathbf{H} \preceq \mathbf{H}'$, then $Q(\mathbf{H}, \mathbf{b}, \mathbf{z}) \leq Q(\mathbf{H}', \mathbf{b}, \mathbf{z})$.*
2. *If $\|\mathbf{H} - \mathbf{H}'\|_1 \leq \delta_a$, then $\left|Q(\mathbf{H}, \mathbf{b}, \mathbf{z}) - Q(\mathbf{H}', \mathbf{b}, \mathbf{z})\right| \leq \frac{1}{2}\delta_a \|\mathbf{z}\|_\infty^2$.*
3. *We have $\left|Q(\mathbf{H}, \mathbf{b}, \mathbf{z}) - Q(\mathbf{H}, \mathbf{b}', \mathbf{z})\right| = \left|\langle \mathbf{b} - \mathbf{b}', \mathbf{z}\rangle\right| \leq \|\mathbf{b} - \mathbf{b}'\|_1 \|\mathbf{z}\|_\infty$.*

Proof of Theorem 3.5. We follow the proof of [12, Thm. 3.4], and use their implementation of a k-oracle $\mathcal{A}$ for $k = O(\log n)$, as detailed in Theorem 3.4. That is, $\mathcal{A}$ takes as input an SDD matrix $\mathbf{H}$ with $\widetilde{O}(m)$ non-zero entries (off-diagonal entries ≤ 0) and a vector $\mathbf{b}$, and outputs a vector $\mathbf{z}$ such that $\|\mathbf{z}\|_\infty \leq k$ and

$$Q(\mathbf{H}, \mathbf{b}, \mathbf{z}) \leq \frac{1}{2} \inf_{\|\mathbf{z}'\|_\infty \leq 1} Q(\mathbf{H}, \mathbf{b}, \mathbf{z}').$$

Then for $\mathbf{x}' = \mathbf{x} + \frac{1}{k}\Delta, \quad \Delta = \mathcal{A}\left(\frac{4e^2}{3k^2}\mathbf{H}, \frac{1}{k}\mathbf{b}\right)$ we have

$$\begin{aligned} Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \frac{1}{k}\Delta\right) &= Q\left(\frac{4e^2}{3k^2}\mathbf{H}, \frac{1}{k}\mathbf{b}, \Delta\right) \leq \frac{1}{2} \inf_{\|z\|_\infty \leq 1} Q\left(\frac{4e^2}{3k^2}\mathbf{H}, \frac{1}{k}\mathbf{b}, \mathbf{z}\right) \\ &= \frac{1}{2} \inf_{\|z\|_\infty \leq 1} Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \mathbf{z}/k\right) = \frac{1}{2} \inf_{\|z\|_\infty \leq \frac{1}{k}} Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \mathbf{z}\right). \end{aligned}$$

Note that the second-order robustness of g implies that for $\tilde{\mathbf{x}} \in \mathbb{R}^n$ with $\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty \leq 1$, we have quadratic lower and upper bounds

$$Q\left(\frac{1}{e^2}\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \tilde{\mathbf{x}} - \mathbf{x}\right) \leq g(\tilde{\mathbf{x}}) - g(\mathbf{x}) \leq Q\left(e^2 \nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \tilde{\mathbf{x}} - \mathbf{x}\right). \tag{B.1}$$

The remainder of the proof is structured as follows. We first compare quadratics involving $\nabla^2 g(\mathbf{x})$ and $\nabla g(\mathbf{x})$ to quadratics involving the approximations $\mathbf{H}$ and $\mathbf{b}$ in Equations (B.2) and (B.3). Using these estimates we then obtain a local progress bound over an ℓ_∞-ball of radius $1/k$, see Equation (B.4). Finally, we convert this local bound into a more global estimate.

The properties of the approximate Hessian and gradient guarantee that

$$\begin{aligned} &Q\left(e^2 \nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \tilde{\mathbf{x}} - \mathbf{x}\right) \\ &\leq Q\left(e^2 \nabla^2 g(\mathbf{x}), \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + \delta \\ &= Q\left(e^2 \mathbf{H}_m', \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(e^2 \mathbf{H}_a', \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle + \delta \\ &\leq Q\left(\frac{4e^2}{3}\mathbf{H}_m, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(e^2 \mathbf{H}_a, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + \frac{e^2}{2}\delta_a \|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle + \delta \\ &\leq Q\left(\frac{4e^2}{3}\mathbf{H}_m, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(\frac{4e^2}{3}\mathbf{H}_a, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + \frac{e^2}{2}\delta_a \|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle + \delta \\ &= Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + \frac{e^2}{2}\delta_a \|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 + \delta. \end{aligned} \tag{B.2}$$

Furthermore, we also have the upper bound

$$\begin{aligned}
&Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) \\
&= Q\left(\frac{4e^2}{3}\mathbf{H}_m, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(\frac{4e^2}{3}\mathbf{H}_a, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle \\
&\leq Q\left(2e^2\mathbf{H}'_m, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(2e^2\mathbf{H}_a, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle \\
&\leq Q\left(2e^2\mathbf{H}'_m, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(2e^2\mathbf{H}'_a, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + e^2\delta_a\|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle \\
&\leq Q\left(2e^2\mathbf{H}'_m, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + Q\left(2e^2\mathbf{H}'_a, \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + e^2\delta_a\|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 - \langle \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\rangle \\
&= Q\left(2e^2\nabla^2 g(\mathbf{x}), \mathbf{b}, \tilde{\mathbf{x}} - \mathbf{x}\right) + e^2\delta_a\|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 \\
&\leq Q\left(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \tilde{\mathbf{x}} - \mathbf{x}\right) + e^2\delta_a\|\tilde{\mathbf{x}} - \mathbf{x}\|_\infty^2 + \delta. \qquad \text{(B.3)}
\end{aligned}$$

Let $\mathbf{v}_L$ and $\mathbf{v}_U$ be the minimizers of quadratics over the ℓ_∞-ball of radius $1/k$:

$$\mathbf{v}_L = \operatorname*{argmin}_{\|\mathbf{v}\|_\infty \leq 1/k} Q(\frac{1}{e^2}\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}), \quad \mathbf{v}_U = \operatorname*{argmin}_{\|\mathbf{v}\|_\infty \leq 1/k} Q(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}).$$

Then by the guarantees of the k-oracle, we have

$$\begin{aligned}
Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \frac{1}{k}\Delta\right) &\leq \frac{1}{2}\inf_{\|\mathbf{v}\|_\infty \leq 1/k} Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \mathbf{v}\right) \\
&\leq \frac{1}{2}\inf_{\|\mathbf{v}\|_\infty \leq 1/k} (Q\left(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}\right) + e^2\delta_a\|\mathbf{v}\|_\infty^2 + \delta) \\
&\leq \frac{1}{2}Q\left(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}_U\right) + \frac{e^2\delta_a}{2k^2} + \frac{1}{2}\delta,
\end{aligned}$$

where the second inequality uses Equation (B.3), and the norm bounds $\|\mathbf{v}\|_\infty \leq 1/k \leq 1$ (to apply the inequality). Using the quadratic upper bound from Equation (B.1) on $g(\mathbf{x} + \frac{1}{k}\Delta) - g(\mathbf{x})$ and Equation (B.2), this yields

$$\begin{aligned}
g(\mathbf{x} + \frac{1}{k}\Delta) - g(\mathbf{x}) \leq Q(e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \frac{1}{k}\Delta) &\leq Q\left(\frac{4e^2}{3}\mathbf{H}, \mathbf{b}, \frac{1}{k}\Delta\right) + \frac{e^2}{2}\delta_a + \delta \\
&\leq \frac{1}{2}Q\left(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}_U\right) + \frac{e^2\delta_a}{k^2} + \frac{3}{2}\delta,
\end{aligned}$$

We can then further upper bound this using

$$Q\left(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}_U\right) \leq Q\left(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \frac{\mathbf{v}_L}{2e^4}\right) = \frac{1}{2e^4}Q\left(\frac{1}{e^2}\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}_L\right)$$

where the inequality uses that $\mathbf{v}_U = \operatorname{argmin}_{\|v\|_\infty \leq 1/k} Q(2e^2\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), v)$ and $\|\mathbf{v}_L\|_\infty \leq 1/k$. Collecting estimates, we obtain

$$g(\mathbf{x} + \frac{1}{k}\Delta) - g(\mathbf{x}) \leq \frac{1}{4e^4}Q\left(\frac{1}{e^2}\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \mathbf{v}_L\right) + \frac{e^2\delta_a}{k^2} + \frac{3}{2}\delta. \qquad \text{(B.4)}$$

We now convert this to a more global estimate. Let $\mathbf{x}^*$ be a global minimizer of g. Set $\mathbf{y} = \mathbf{x} + \frac{1}{\max(kR_\infty, 1)}(\mathbf{x}^* - \mathbf{x})$, so that $\|\mathbf{y} - \mathbf{x}\|_\infty \leq \frac{1}{k}$. For the lower bound

$$g_L(\tilde{\mathbf{x}}) = g(\mathbf{x}) + Q(\frac{1}{e^2}\nabla^2 g(\mathbf{x}), \nabla g(\mathbf{x}), \tilde{\mathbf{x}} - \mathbf{x})$$

on $g(\tilde{\mathbf{x}})$ we see that $g_L(\mathbf{x}+\mathbf{v}_L) \leq g_L(\mathbf{y}) \leq g(\mathbf{y})$ since $\mathbf{x}+\mathbf{v}_L$ minimizes $g_L \leq g$ over the ℓ_∞-ball of radius $1/k$ around $\mathbf{x}$. By convexity of g we get

$$g(\mathbf{y}) = g(\mathbf{x} + \frac{1}{\max(kR_\infty, 1)}(\mathbf{x}^* - \mathbf{x})) \leq (1 - \frac{1}{\max(kR_\infty, 1)})g(\mathbf{x}) + \frac{1}{\max(kR_\infty, 1)}g(\mathbf{x}^*)$$

so

$$g(\mathbf{x}) - g_L(\mathbf{x}+\mathbf{v}_L) \geq g(\mathbf{x}) - g(\mathbf{y}) \geq \frac{1}{\max(kR_\infty, 1)}(g(\mathbf{x}) - g(\mathbf{x}^*)).$$

Using this estimate in Equation (B.4), this gives

$$g(\mathbf{x}) - g(\mathbf{x} + \frac{1}{k}\Delta) \geq \frac{1}{4e^4 \max(kR_\infty, 1)}(g(\mathbf{x}) - g(\mathbf{x}^*)) - (\frac{e^2\delta_a}{k^2} + \frac{3}{2}\delta),$$

which after rearranging and rewriting $\mathbf{x}' = \mathbf{x} + \frac{1}{k}\Delta$ reads

$$g(\mathbf{x}') - g(\mathbf{x}^*) \leq \left(1 - \frac{1}{4e^4 \max(kR_\infty, 1)}\right)(g(\mathbf{x}) - g(\mathbf{x}^*)) + \frac{e^2\delta_a}{k^2} + \frac{3}{2}\delta.$$ ◀

B.2 Approximating the Hessian of the regularized potential

Proof of Lemma 3.6. The first point is easy to verify, as is the second point (the regularization term is always positive). For the third point, suppose we have a $\mathbf{z}$ such that $\tilde{f}(\mathbf{z}) \leq \tilde{f}(0)$. Then

$$\frac{\varepsilon^2}{ne^B}\left(\sum_i (e^{x_i} + e^{-x_i}) + \sum_j (e^{y_j} + e^{-y_j})\right) \leq f(\mathbf{0}) - f(\mathbf{z}) + \frac{\varepsilon^2}{ne^B}\cdot 4n \leq \ln(1/\mu) + \frac{4\varepsilon^2}{e^B}. \quad \text{(B.5)}$$

where the last inequality follows from the potential bound $f(\mathbf{0}) - f^* \leq \ln(1/\mu)$ (which depends on $\|\mathbf{A}\|_1 \leq 1$; in general the upper bound is $\|\mathbf{A}\|_1 - 1 + \ln(1/\mu)$). Since each of the regularization terms is positive, we may restrict ourselves to a single term and see that $e^{x_i} + e^{-x_i} \leq \frac{e^B n \ln(1/\mu)}{\varepsilon^2} + 4n$, from which we may deduce

$$|x_i| \leq \ln\left(\frac{e^B n \ln(1/\mu)}{\varepsilon^2} + 4n\right) = B + \ln\left(\frac{n\ln(1/\mu)}{\varepsilon^2} + \frac{4n}{e^B}\right) \leq B + \ln\left(\frac{n\ln(1/\mu)}{\varepsilon^2} + 4n\right),$$

where the last inequality uses $e^B \geq 1$ (recall $B > 0$). The same upper bound holds for $|y_j|$.

For the last point, note that if $\mathbf{z}_\varepsilon = (\mathbf{x}, \mathbf{y})$, then $e^{x_i} + e^{-x_i} \leq 2e^B$ and similarly for y, so

$$\tilde{f}(\mathbf{z}_\varepsilon) \leq f(\mathbf{z}_\varepsilon) + \frac{\varepsilon^2}{ne^B}\cdot 4ne^B = f(\mathbf{z}_\varepsilon) + 4\varepsilon^2 \leq f^* + 5\varepsilon^2.$$

If such a $\mathbf{z}_\varepsilon$ exists, then $f^* \leq \tilde{f}^* \leq \tilde{f}(\mathbf{z}_\varepsilon) \leq f^* + 5\varepsilon^2$. ◀

Tight Bounds for Counting Colorings and Connected Edge Sets Parameterized by Cutwidth

Carla Groenland
Utrecht University, The Netherlands

Isja Mannens
Utrecht University, The Netherlands

Jesper Nederlof
Utrecht University, The Netherlands

Krisztina Szilágyi
Utrecht University, The Netherlands

Abstract

We study the fine-grained complexity of counting the number of colorings and connected spanning edge sets parameterized by the cutwidth and treewidth of the graph. While decompositions of small treewidth decompose the graph with small vertex separators, decompositions with small cutwidth decompose the graph with small *edge* separators.

Let $p, q \in \mathbb{N}$ such that p is a prime and $q \geq 3$. We show:

- If p divides $q-1$, there is a $(q-1)^{\mathrm{ctw}} n^{O(1)}$ time algorithm for counting list q-colorings modulo p of n-vertex graphs of cutwidth ctw. Furthermore, there is no $\varepsilon > 0$ for which there is a $(q-1-\varepsilon)^{\mathrm{ctw}} n^{O(1)}$ time algorithm that counts the number of list q-colorings modulo p of n-vertex graphs of cutwidth ctw, assuming the Strong Exponential Time Hypothesis (SETH).
- If p does not divide $q-1$, there is no $\varepsilon > 0$ for which there exists a $(q-\varepsilon)^{\mathrm{ctw}} n^{O(1)}$ time algorithm that counts the number of list q-colorings modulo p of n-vertex graphs of cutwidth ctw, assuming SETH.

The lower bounds are in stark contrast with the existing $2^{\mathrm{ctw}} n^{O(1)}$ time algorithm to compute the chromatic number of a graph by Jansen and Nederlof [Theor. Comput. Sci.'18].

Furthermore, by building upon the above lower bounds, we obtain the following lower bound for counting connected spanning edge sets: there is no $\varepsilon > 0$ for which there is an algorithm that, given a graph G and a cutwidth ordering of cutwidth ctw, counts the number of spanning connected edge sets of G modulo p in time $(p-\varepsilon)^{\mathrm{ctw}} n^{O(1)}$, assuming SETH. We also give an algorithm with matching running time for this problem.

Before our work, even for the treewidth parameterization, the best conditional lower bound by Dell et al. [ACM Trans. Algorithms'14] only excluded $2^{o(\mathrm{tw})} n^{O(1)}$ time algorithms for this problem.

Both our algorithms and lower bounds employ use of the matrix rank method, by relating the complexity of the problem to the rank of a certain "compatibility matrix" in a non-trivial way.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases connected edge sets, cutwidth, parameterized algorithms, colorings, counting modulo p

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.36

Related Version *Full Version*: https://arxiv.org/abs/2110.02730

Funding All authors are supported by the project CRACKNP that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 853234).

Acknowledgements We thank the anonymous reviews for their detailed comments.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 36; pp. 36:1–36:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A popular topic of interest in (fine-grained) algorithmic research is to determine the decomposability of NP-hard problems in easier subproblems. A natural decomposition strategy is often implied by decomposing the solution into sub-solutions induced by a given decomposition of the input graph such as tree decompositions, path decompositions, or tree depth decompositions, independent of the problem to be solved. However, the efficiency of such a decomposition can wildly vary per computational problem. Recently, researchers developed tools that allow them to get a precise understanding of this efficiency : non-trivial algorithmic tools (such as convolutions and the cut-and-count method [9, 25]) were developed to give algorithms that have an optimal running time conditioned on hypotheses such as the Strong Exponential Time Hypothesis (SETH) [16]. While the efficiency of such decompositions has been settled for most decision problems parameterized by treewidth, many other interesting settings remain elusive. Two of them are *cutwidth* and *counting problems*.

The *cutwidth* of an ordering of the vertices of the graph is defined as the maximum number of edges with exactly one endpoint in a prefix of the ordering (where the maximum is taken over all prefixes of the ordering). The *cutwidth* of a graph is defined to be the minimum width over all its cutwidth orderings. Cutwidth is very similar to pathwidth, except that cutwidth measures the number of edges of a cut, while the pathwidth measures the number of *endpoints* of edges over the cut. Thus the cutwidth of a graph is always larger than its pathwidth. But for some problems a decomposition scheme associated with a cutwidth ordering of cutwidth k can be used much more efficiently than a decomposition of pathwidth k. A recent example of such a problem is the q-coloring problem:[1] While there is a $(q-\varepsilon)^{\mathrm{pw}}$ lower bound [21] assuming SETH, there is a $2^{\mathrm{ctw}}n^{O(1)}$ time randomized algorithm [18].

Counting problems pose an interesting challenge if we want to study their decomposability. Counting problems are naturally motivated if we are interested in any statistic rather than just existence of the solutions space. While often a counting problem behaves very similarly to its decision version (as in, the dynamic programming approach can be fairly directly extended to solve the counting version as well), for some problems there is a rather puzzling increase in complexity when going from the decision version to the counting version. [2]

One of the most central problems in counting complexity is the evaluation of the *Tutte polynomial*. The strength of this polynomial is that it expresses all graph invariants that can be written as a linear recurrence using only the edge deletion and contraction operation [23], and its evaluations specialize to a diverse set of parameters ranging from the number of forests, nowhere-0 flows, q-colorings and spanning connected edge sets.

An interesting subdirection within counting complexity that is in between the decision and counting version and that we will also address in this paper is *modular counting*, where we want to count the number of solutions modulo a number p. This is an interesting direction since the complexity of the problem at hand can wildly vary for different p (see [24] for a famous example), but in the setting of this paper it is also naturally motivated: For example, the cut-and-count method achieves the fastest algorithms for several decision problems by actually solving the modular counting variant instead.

[1] Recall that a q-coloring is a mapping from the vertices of the graph to $\{1,\dots,q\}$ such that every two adjacent vertices receive distinct colors, and the q-coloring problem asks whether a q-coloring exists.

[2] Two examples herein are detecting/counting perfect matchings (while the decision version is in P, the counting version can not be solved in time $(2-\varepsilon)^{\mathrm{tw}}n^{O(1)}$ for any $\varepsilon > 0$ assuming the SETH [7]) and Hamiltonian cycles (while the decision version can be solved in $(2+\sqrt{2})^{\mathrm{pw}}$ time [8], the counting version can not be solved in time $(6-\varepsilon)^{\mathrm{tw}}n^{O(1)}$ for any $\varepsilon > 0$ assuming the SETH [6]).

1.1 Our results

In this paper we study the complexity of two natural hard (modular) counting problems: Counting the number of q-colorings of a graph and counting the number of spanning connected edge sets, parameterized by the cutwidth of the graph.

Counting Colorings. Let G be a graph and suppose that for each $v \in V$ we have an associated list $L(v) \subseteq \{1, \ldots, q\}$. A *list q-coloring* is a coloring c of G such that $c(v) \in L(v)$ for each $v \in V$. Two colorings are *essentially distinct* if they cannot be obtained from each other by permuting the color classes. Since the number of essentially distinct colorings is $q!$ times the number of distinct colorings (assuming the chromatic number of the graph is q), counting colorings modulo p may become trivial if $p \leq q$. For this reason, we focus on counting essentially distinct colorings in our lower bounds.

In this paper, we will focus on counting list q-colorings modulo a prime number p. Our main theorem reads as follows:

▶ **Theorem 1.** *Let $p, q \in \mathbb{N}$ with p prime and $q \geq 3$.*

- *If p divides $q-1$, then there is a $(q-1)^{\text{ctw}} n^{O(1)}$ time algorithm for counting list q-colorings modulo p of n-vertex graphs of cutwidth* ctw*. Furthermore, there is no $\varepsilon > 0$ for which there exists a $(q-1-\varepsilon)^{\text{ctw}} n^{O(1)}$ time algorithm that counts the number of essentially distinct q-colorings modulo p in time $(q-1-\varepsilon)^{\text{ctw}} n^{O(1)}$, assuming SETH.*
- *If p does not divide $q-1$, there is no $\varepsilon > 0$ for which there exists a $(q-\varepsilon)^{\text{ctw}} n^{O(1)}$ time algorithm that counts the number of essentially distinct q-colorings modulo p, assuming SETH.*

Thus, we show that under the cutwidth parameterization, the (modular) counting variant of q-coloring is much harder than the decision, as the latter can be solved in $2^{\text{ctw}} n^{O(1)}$ time with a randomized algorithm [18]. Additionally, we show there is a curious jump in complexity based on whether p divides $q-1$ or not: Since our bounds are tight, this jump is inherent to the problem and not an artifact of our proof.

The proof strategy of all items of Theorem 1 relates the complexity of the problems to a certain *compatibility matrix*. This is a Boolean matrix that has its rows and columns indexed by partial solutions, and has a 1 if and only if the corresponding partial solutions combine into a global solution. In previous work, it was shown that the rank of this matrix can be used to design both algorithms [4, 8, 18, 22] and lower bounds [6, 8].

With this in mind, the curious jump can intuitively be explained as follows. Consider the base case where the graph is a single edge and we decompose a (list) q-coloring into the two colorings induced on the vertices. The compatibility matrix corresponding to this decomposition is the complement of an $q \times q$ identity matrix. This matrix has full rank if p does not divide $q-1$ and it has rank $q-1$ otherwise. We believe this is a very clean illustration of the rank based methods, since it explains a curious gap that would be rather mysterious without the rank based perspective.

Connected Spanning Edge Sets and Tutte polynomial. We say that $X \subseteq E$ is a *connected spanning edge set* if $G[X]$ is connected and every vertex is adjacent to an edge in X. Our second result is about counting the number of such sets. This problem is naturally motivated: It gives the probability that a random subgraph remains connected, and is an important special case of the Tutte polynomial. We determine the complexity of counting connected spanning edge sets by treewidth and cutwidth by giving matching lower and upper bounds:

▶ **Theorem 2.** *Let p be a prime number. There is an algorithm that counts the number of connected edge sets modulo p of n-vertex graphs of treewidth* tw *in time $p^{\mathrm{tw}}n^{O(1)}$.*

Furthermore, there is no $\varepsilon > 0$ for which there is an algorithm that counts the number of spanning connected edge sets modulo p of n-vertex graphs of cutwidth ctw *in time $(p-\epsilon)^{\mathrm{ctw}}n^{O(1)}$, assuming SETH.*

Note that before our work, even for the treewidth parameterization, the best conditional lower bound by Dell et al. [10] only excluded $2^{o(\mathrm{tw})}n^{O(1)}$ time algorithms for this problem.

While the algorithm follows relatively quickly by using a cut-and-count type dynamic programming approach, obtaining the lower bound is much harder.

In fact, for related counting variants of connectivity problems such as counting the number of Hamiltonian cycles or Steiner trees, $2^{O(\mathrm{tw})}n^{O(1)}$ time algorithms do exist. So one may think that connected spanning edge sets can be counted in a similar time bound. But in Theorem 2 we show that this is not the case (by choosing p arbitrarily large).

To prove the lower bound, we make use of an existing formula for the Tutte polynomial that relates the number of connected spanning edge sets to the number of essentially distinct colorings, and subsequently apply Theorem 1.

Organization. The rest of the paper is organized as follows: in Section 2 we introduce the notation that will be used throughout the paper and define the color compatibility matrix. In Section 3 we prove the upper bound for $\#q$-coloring modulo p. Section 4 contains the results about lower bounds. We conclude the paper by discussing directions for further research. The appendix contains the proofs omitted from previous sections.

1.2 Related work

Coloring. Counting the number of colorings of a graph is known to be $\#P$-complete, even for special classes of graphs such as triangle free regular graphs [14]. Björklund and Husfeldt [2] and Koivisto [20] gave a $2^n n^{O(1)}$ algorithm for counting q-colorings, and a more general $2^n n^{O(1)}$ time algorithm even evaluates any point of the Tutte polynomial [3].

A q-coloring of a graph G is a special case of *H-coloring*, i.e. a homomorphism from G to a given graph H. Namely, q-colorings correspond to homomorphisms from G to K_q, i.e. K_q-colorings. Dyer and Greenhill [11] showed that counting the number of H-colorings is $\#P$-complete unless H is one of the few exceptions (an independent set, a complete graph with loops on every vertex or a complete bipartite graph). Kazeminia and Bulatov [19] classified the hardness of counting H-colorings modulo a prime p for square-free graphs H.

Methods. Our approach makes use of the rank based method, and in particular the so-called color compatibility matrix introduced in [18]. This matrix tells us whether we can "combine" two colorings. In [18], the authors studied the rank of a different matrix with the same support as the color compatibility matrix, whereas in this paper we use the rank directly. The rank based method has been used before only once for an algorithm for a counting problem in [8] and only once for a lower bound for a counting problem in [6].

The Tutte polynomial $T(G; x, y)$ is a graph polynomial in two variables which describes how G is connected. In particular, calculating $T(G; x, y)$ at specific points gives us the number of subgraphs of G with certain properties: $T(G; 2, 1)$ is equal to the number of forests in G, $T(G; 1, 1)$ is the number of spanning forests, $T(G; 1, 2)$ counts the number of spanning connected subgraphs etc. We will use the properties of the Tutte polynomial to give a lower bound on the complexity of counting spanning connected edge sets.

2 Preliminaries

In this section, we introduce the notation that will be used throughout the paper.

2.1 Notation and standard definitions

For integers a, b, we write $[a, b] = \{a, a+1, \dots, b\}$ for the integers between a and b, and for a natural number n we short-cut $[n] = [1, n] = \{1, \dots, n\}$. Throughout the paper, p will denote a prime number and $\mathbb{F}_p$ the finite field of order p. We will use $a \equiv_p b$ to denote that a and b are congruent modulo p, i.e. that p divides $a - b$. We write $\mathbb{N} = \{1, 2, \dots\}$ for the set of natural numbers.

For a function $f : A \to \mathbb{Z}$ (where A is any set), we define the support of f as the set $\text{supp}(f) = \{a \in A \ : \ f(a) \neq 0\}$. For $B \subseteq A$, the function $f|_B : B \to \mathbb{Z}$ is defined as $f|_B(b) = f(b)$ for all $b \in B$.

In this paper, all graphs will be undirected and simple. Given a graph $G = (V, E)$ and a vertex $v \in V$, we denote by $N(v)$ the open neighbourhood of v, i.e. the set of all vertices adjacent to v. We often use n for the number of vertices of G, and denote the cutwidth of G by ctw. We sometimes write $V(G)$ for the vertex set of the graph G.

Note that, if G is not connected, we can count the number of q-colorings in each connected component and multiply them to get the total number of q-colorings of G. Therefore, we may assume that G is connected.

Given a graph $G = (V, E)$, and lists $L : V \to 2^{[q]}$, a *list q-coloring* of G is a coloring $c : V \to [q]$ of its vertices such that $c(u) \neq c(v)$ for all edges uv and $c(v) \in L(v)$ for all vertices v. We will often abbreviate "list q-coloring" to "coloring". For a subset $B \subseteq V(G)$, we will use the abbreviation $c(B) = \{c(v) : v \in B\}$.

Cutwidth and treewidth are graph parameters which are often used in parameterized complexity. Informally, treewidth describes how far a graph is from being a tree. The cutwidth is defined as follows.

▶ **Definition 3.** *The* cutwidth *of a graph G is the smallest k such that its vertices can be arranged in a sequence $v_1, \dots, v_n$ such that for every $i \in [n-1]$, there are at most k edges between $\{v_1, \dots, v_i\}$ and $\{v_{i+1}, \dots, v_n\}$.*

We recall the definition of Tutte polynomial.

▶ **Definition 4.** *For a graph G, we denote by $T(G; x, y)$ the* Tutte polynomial *of G evaluated at the point (x, y). If G has no edges we have $T(G; x, y) = 1$. Otherwise we have*

$$T(G; x, y) = \sum_{A \subseteq E(G)} (x-1)^{r(E)-r(A)}(y-1)^{|A|-r(A)}$$

where $r(A) = |V(G)| - k(A)$ indicates the rank of the edge set A and $k(A)$ indicates the number of connected components of (V, A).

Note that $T(G; 1, 2)$ is exactly the number of spanning connected edge sets.

We denote the counting version of a problem by using the prefix #, and the counting modulo p version of by using $\#_p$ (e.g. $\#_p$SAT, $\#_p$CSP).

2.2 The color compatibility matrix and its rank

Given a subset $A \subseteq V$, we use $\text{col}_L(A)$ to denote the set of all list q-colorings of $G[A]$. If it is clear which lists are used, we omit the subscript.

It is often useful to color parts of the graph separately, and then "combine" those colorings. If two colorings can be combined without conflicts, we call them compatible:

▶ **Definition 5.** *For subsets $A, B \subseteq V$ and colorings $x \in \mathrm{col}(A)$, $z \in \mathrm{col}(B)$, we say that x and z are* compatible*, written $x \sim z$, if*

- *$x(v) = z(v)$ for all $v \in A \cap B$, and*
- *$x(u) \neq z(v)$ for all $uv \in E$, where $u \in A$ and $v \in B$.*

For a set of colorings $\mathcal{S} \subseteq \mathrm{col}(B)$, we write $\mathcal{S}[x]$ for the set of colorings $y \in \mathcal{S}$ that are compatible with x.

If $x \sim z$, then we define $x \cup z$ as the q-list coloring of $G[A \cup B]$ such that $(x \cup z)(a) = x(a)$ for all $a \in A$ and $(x \cup z)(b) = z(b)$ for all $b \in B$. This is well-defined by the definition above.

A key definition for this paper is the following.

▶ **Definition 6.** *Let $(X \cup Y, E)$ be a bipartite graph and q a natural number. The qth* color compatibility matrix *M is indexed by all q-colorings of X and Y, with*

$$M[x, y] = \begin{cases} 1, & \text{if } x \sim y, \\ 0, & \text{otherwise,} \end{cases}$$

for $x \in \mathrm{col}(X)$ and $y \in \mathrm{col}(Y)$.

We denote the color compatibility matrix indexed by all q-colorings associated with the bipartite graph that is matching on t vertices by J_t, and short-hand $J := J_1$.

We will show that, if p divides $q - 1$, we can count all q-list colorings modulo p more quickly due to the following bound on the rank of the color compatibility matrices.

▶ **Lemma 7.** *Let p be a prime, q a natural number and let $G = (X \cup Y, E)$ be a bipartite graph with qth color compatibility matrix M. Then the rank of M over $\mathbb{F}_p$ satisfies*

$$\mathrm{rank}_p(M) \leq \begin{cases} (q-1)^{|E|} & \text{if } p \text{ divides } q - 1, \\ q^{|E|} & \text{otherwise.} \end{cases}$$

Moreover, equality is achieved if G is a perfect matching.

The proof can be found in the full version of our paper [15].

In particular, J_t is invertible mod p if and only if p does not divide $q - 1$.

3 Algorithm for #q-coloring modulo p

In this section we prove the first part of the first item of Theorem 1:

▶ **Theorem 8.** *Let G be a graph with n vertices and cutwidth* ctw*. Given an integer $q \geq 3$ and a prime p that divides $q - 1$, there is an $(q-1)^{\mathrm{ctw}} n^{O(1)}$ algorithm for counting list q-colorings modulo p.*

3.1 Definitions and overview

We first introduce some additional notation and definitions needed in this section. Let q be an integer and let p be a prime that divides $q - 1$. We are given a graph $G = (V, E)$ with the cutwidth ordering $v_1, \ldots, v_n$ of the vertices. Without loss of generality, we may assume that G is connected. We write $G_i = G[\{v_1, \ldots, v_i\}]$ and

$$L_i = \{v \in V(G_i) : vv_j \in E \text{ for some } j > i\}.$$

Note that by definition of cutwidth, $L_i \subseteq L_{i-1} \cup \{v_i\}$ and $|L_i| \leq \mathrm{ctw}$ for all i (since the number of endpoints of a set of edges is upper bounded by the number of edges in the set).

Let $i \in [n]$ be given and write $X_i = L_i \cup \{v_i\}$ for the set of vertices left of the cut that either have an edge in the cut, or are the rightmost vertex left of the cut. We also define $Y_i = \{v_{i+1}, \dots, v_n\} \cap N(X_i)$. Figure 1 illustrates this notation.

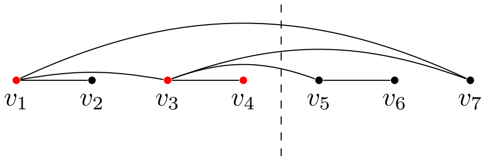


Figure 1 In the above graph, $L_4 = \{v_1, v_3\}$, $X_4 = \{v_1, v_3, v_4\}$ (the red vertices) and $Y_4 = \{v_5, v_7\}$.

Let $T_i[x]$ be the number of extensions of $x \in \mathrm{col}(X_i)$ to a coloring of $G_i = G[\{v_1, \dots, v_i\}]$. Equivalently, $T_i[x]$ gives the number of colorings of G_i that are compatible with x.

A standard dynamic programming approach builds on the following observation.

▶ **Lemma 9** (Folklore). *For $x \in \mathrm{col}(X_i)$,*

$$T_i[x] = \sum_{\substack{z \in \mathrm{col}(X_{i-1}) \\ z \sim x}} T_{i-1}[z].$$

The proof can be found in the full version of our paper [15]. Since $|\mathrm{col}(X_i)|$ may be of size $q^{|X_i|}$, we cannot compute T_i in its entirety within the claimed time bound. The idea of our algorithm is to use the same dynamic programming iteration, but to compute the values of T_i only for a subset $\mathcal{S}'_i \subseteq \mathrm{col}(X_i)$ of the possible colorings which is of significantly smaller size. In fact, we will compute a function $T'_i : \mathcal{S}'_i \to \mathbb{F}_p$ that does not necessarily agree with T_i on $\mathcal{S}'_i$. The important property that we aim to maintain is that T'_i carries the "same information" about the number of colorings modulo p as T_i does. This is formalised below.

▶ **Definition 10.** *Let $H = (X \cup Y, E)$ be a bipartite graph with color compatibility matrix M. Let $T, T' : \mathrm{col}(X) \to \mathbb{F}_p$. We say T' is an M-*representative *of T if*

$$\sum_{x \in \mathrm{col}(X)} M[x,y]T[x] \equiv_p \sum_{x \in \mathrm{col}(X)} M[x,y]T'[x] \text{ for all } y \in \mathrm{col}(Y).$$

In other words, T' is an M-representative of T if $M^\top \cdot T \equiv_p M^\top \cdot T'$.

Above we left the lists and the integer q implicit. We recall that the color compatibility matrix has entries $M[x,y] = 1$ if $x \in \mathrm{col}(X)$ and $y \in \mathrm{col}(Y)$ are compatible, and $M[x,y] = 0$ otherwise. Let $i \in [n-1]$ be given. Let M_i be the color compatibility matrix of the bipartite graph given by the edges between X_i and Y_i.

Then for $y \in \mathrm{col}(Y_i)$,

$$\sum_{x \in \mathrm{col}(X_i)} M_i[x,y]T_i[x]$$

gives the number of colorings of G_i compatible with y. If we can compute T'_{n-1} that is an M_{n-1}-representative of T_{n-1}, then by Lemma 9 we can compute the number of q-list colorings of the graph (modulo p) as

$$\sum_{y \in \mathrm{col}(G[v_n])} \sum_{x \in \mathrm{supp}(T'_{n-1})} M_{n-1}[x,y]T'_{n-1}[x].$$

It is an exercise in linear algebra to show that there always exists a T' that M-represents T with $|\mathrm{supp}(T')| \le \mathrm{rank}(M)$. We also need to make sure that we can actually compute this T' within the desired time complexity and therefore reduce the support in a slightly more complicated fashion in Section 3.2. We then prove an analogue of Lemma 9 in Section 3.3, and describe our final algorithm in Section 3.4.

3.2 Computing a reduced representative

In this subsection, we show how to find a function T' that M-represents T, while decreasing an upper bound on the size of the support of the function.

▶ **Definition 11.** *For a function $f : \mathrm{col}(X) \to \mathbb{F}_p$ we say that $r \in X$ is a* reduced vertex *if $f(c) = 0$ whenever $c(r) = q$.*

The link between reduced vertices and the support of $T : \mathrm{col}(X) \to \mathbb{F}_p$ is explained as follows. If R is a set of reduced vertices of T, then we can compute a set of colorings containing the support of T of size at most $(q-1)^{|R|}q^{|X|-|R|}$. Indeed, we may restrict to the colorings that do not assign the color q to any vertex in R.

The following result allows us to turn vertices of degree 1 in H into reduced vertices. The assumption that the vertex has degree 1 will be useful in proving the result because it implies the associated compatibility matrix can be written as a Kronecker product with J_q and another matrix.

▶ **Lemma 12.** *There is an algorithm* ***Reduce*** *that, given a bipartite graph H with parts X, Y and associated color compatibility matrix M, a function $T : \mathrm{col}(X) \to \mathbb{F}_p$ with reduced vertices $R \subseteq X$ and a vertex $v \in X \setminus R$ of degree 1 in H, outputs a function $T' : \mathrm{col}(X) \to \mathbb{F}_p$ with reduced vertices $R \cup \{v\}$ that is an M-representative of T. The run time is in $O((q-1)^{|R|}q^{|X|-|R|})$.*

The proof is given in Appendix A. We say that a function $T : \mathrm{col}(X) \to \mathbb{F}_p$ is *fully reduced* if every vertex $v \in X$ of degree 1 is a reduced vertex of T. In order to keep the running time low, we will ensure that R is relatively large whenever we apply Lemma 12.

3.3 Computing T'_i from T'_{i-1}

Recall that $T_i[x]$ gives the number of colorings of G_i that are compatible with $x \in \mathrm{col}(X_i)$ and that M_i is the color compatibility matrix of the bipartite graph between X_i and Y_i (corresponding to the ith cut).

▶ **Lemma 13.** *Let $i \in [n-1]$. Suppose that T'_{i-1} is an M_{i-1}-representative of T_{i-1} and that T'_{i-1} is fully reduced. Given T'_{i-1} and a set R_{i-1} of reduced vertices for T'_{i-1}, we can compute a function T'_i that is an M_i-representative of T_i in time $O((q-1)^{|R_{i-1}|}q^{|X_{i-1}|-|R_{i-1}|+1})$, along with a set R_i of reduced vertices for T'_i such that $|X_i \setminus R_i| \leq (\mathrm{ctw} - |R_i|)/2 + 1$.*

Proof. Let $i \in [n-1]$ and let T'_{i-1} be M_{i-1}-representative of T_{i-1} and fully reduced, with R_{i-1} a set of reduced vertices for T'_{i-1}. We need to compute (in time $O((q-1)^{|R_{i-1}|}q^{|X_{i-1}|-|R_{i-1}|+1})$) a function T'_i that is M_i-representative of T_i, along with a set R_i of reduced vertices for T'_i, such that $|X_i \setminus R_i| \leq (\mathrm{ctw} - |R_i|)/2 + 1$.

We will work over $\mathbb{F}_p$ during this proof, in particular abbreviating $\equiv_p$ to $=$. Analogous to Lemma 9, we define, for $x \in \mathrm{col}(X_i)$,

$$T'_i[x] = \sum_{\substack{z \in \mathrm{col}(X_{i-1}) \\ z \sim x}} T'_{i-1}[z]. \tag{1}$$

Note that

$$\sum_{\substack{z \in \mathrm{col}(X_{i-1}) \\ z \sim x}} T'_{i-1}[z] = \sum_{\substack{z \in \mathrm{supp}(T'_{i-1}) \\ z \sim x}} T'_{i-1}[z].$$

We compute T'_i from T'_{i-1} as follows. Let

$$\mathcal{S}'_{i-1} = \{c \in \mathrm{col}(X_{i-1}) : c(r) \neq q \text{ for all } r \in R_{i-1}\}.$$

By the definition of reduced vertex, $\mathcal{S}'_{i-1}$ contains the support of T'_{i-1} since T'_{i-1} is fully reduced. Recall that $X_i \setminus \{v_i\} \subseteq X_{i-1}$, so any $x \in \mathrm{col}(X_i)$ is determined if we provide colors for the vertices in $X_{i-1} \cup \{v_i\}$. For a color $c \in [q]$, let $f_c : \{v_i\} \to \{c\}$ be the function that assigns color c to v_i. For each $z \in \mathcal{S}'_{i-1}$, for each $c \in [q]$ for which $z \sim f_c$, we compute

$$x = (z \cup f_c)|_{X_i} \in \mathrm{col}(X_i)$$

and increase $T'_i[x]$ by $T'_{i-1}[z]$ if it has been defined already, and initialise it to $T'_{i-1}[z]$ otherwise. The remaining values are implicitly defined to 0. The running time is as claimed because $|\mathcal{S}'_{i-1}| \leq (q-1)^{|R_{i-1}|} q^{|X_{i-1}|-|R_{i-1}|}$ and $|[q]| \leq q$.

Next, we compute a set R_i of reduced vertices for T'_i. We set $R_i = X_i \setminus (A_i \cup B_i \cup \{v_i\})$, where

$$A_i = \{u \in X_i \setminus \{v_i\} \ : \ |N(u) \cap Y_i| \geq 2\}$$

and

$$B_i = \{u \in X_i \setminus \{v_i\} \ : \ |N(u) \cap Y_i| = 1 \text{ and } uv_i \in E\}.$$

It is easy to see that A_i and B_i are disjoint. Within the $(i-1)$th cut, each vertex in $A_i \cup B_i$ has at least two edges going across the cut, so $|R_i| + 2|A_i| + 2|B_i| \leq \mathrm{ctw}$. Therefore, $|X_i \setminus R_i| \leq (\mathrm{ctw} - |R_i|)/2 + 1$.

We now show that R_i is indeed a set of reduced vertices. Suppose not, and let $r \in R_i$ and $c \in \mathrm{col}(X_i)$ with $c(r) = q$ yet $T'_i[c] \neq 0$. Since $T'_i[c] \neq 0$, there exists $z \in \mathrm{col}(X_{i-1})$ with $z \sim c$ and $T'_{i-1}[z] \neq 0$. By definition $r \in X_i \setminus \{v_i\} \subseteq X_{i-1}$. Moreover, $z(r) = q$ since $z \sim c$ and $c(r) = q$. Therefore r is not reduced for T'_{i-1}. We now show r moreover has degree 1 in the bipartite graph between X_{i-1} and Y_{i-1} (corresponding to the $(i-1)$th cut), contradicting our assumption that T'_{i-1} is fully reduced. Since $r \notin A_i \cup B_i$, it has at most one edge going over the $(i-1)$th cut. Moreover, $r \in X_i \setminus \{v_i\} \subseteq L_i$, and so it has at least one edge to $Y_i \subseteq Y_{i-1}$. So r has exactly one neighbor in Y_{i-1}.

It remains to prove that T'_i is M_i-representative of T_i. This technical part of the proof can be found in the full version of our paper [15]. ◀

3.4 Analysis of final algorithm

We initialize $T_1 = \mathbf{1}$, the all-ones vector. Indeed, each $x \in \mathrm{col}(\{v_1\})$ has a unique extension to G_1 (namely itself). We then repeatedly apply the **Reduce** algorithm from Lemma 12 until we obtain a fully reduced function T'_1 that is an M_1-representative of T_1, with some set of reduced vertices R_1. For $i = 2, \ldots, n$, we repeat the following two steps.

1. Apply Lemma 13 with inputs (T'_{i-1}, R_{i-1}) in order to obtain the vector T'_i that is an M_i-representative of T_i, and a set of reduced vertices R_i for T'_i.
2. While $X_i \setminus R_i$ has a vertex v of degree 1, apply the **Reduce** algorithm from Lemma 12 to (T'_i, R_i), and add v to R_i.

At the end of step 2, we obtain a fully reduced function T'_i that is an M_i-representative of T_i. Moreover, the set R_i of reduced vertices has only increased in size compared to the set we obtained in step 1. We apply Lemma 12 at most $|X_i|$ times in the second step.

We eventually compute T'_{n-1} that is an M_{n-1}-representative of T_{n-1} with a fully reduced set R_{n-1}. We output

$$\sum_{y\in\mathrm{col}(Y_{n-1})}\sum_{x\in\mathrm{col}(X_{n-1})} T'_{n-1}[x]M_{n-1}[x,y].$$

Since T'_{n-1} is an M_{n-1}-representative of T_{n-1}, this gives the number of list colorings of G modulo p. We may compute the expression above efficiently by reducing the second summation to the colorings in

$$\mathcal{S}'_{n-1} = \{c \in \mathrm{col}(X_{n-1}) : c(r) \neq q \text{ for all } r \in R_{n-1}\}.$$

The total running time is now bounded by

$$C\sum_{i=1}^{n-1} |X_i|(q-1)^{|R_i|}q^{|X_i|-|R_i|}$$

for some constant $C > 0$. By Lemma 13, $|X_i \setminus R_i| \leq (\mathrm{ctw} - |R_i|)/2 + 1$ for all $i \in [n-1]$. For $q \geq 3$, $q^{1/2} < q - 1$ and so

$$(q-1)^{|R_i|}q^{|X_i|-|R_i|} \leq q(q-1)^{|R_i|}(q^{1/2})^{\mathrm{ctw}-|R_i|} < q(q-1)^{\mathrm{ctw}}.$$

This shows the total running time is of order $(q-1)^{\mathrm{ctw}}n^{O(1)}$. This finishes the proof of Theorem 8.

4 Lower bounds

There exists an efficient reduction from SAT to the problem $\#_p$SAT of counting the number of satisfying assignments for a given boolean formula modulo p [5]. There also exists a reduction from SAT to CSP(q,r), which preserves the number of solutions [12]. Putting these two together gives a reduction from SAT to $\#_p$CSP(q,r).

In this section we give a reduction from $\#_p$SAT to $\#_p$List q-Coloring, the problem of counting the number of valid list q-colorings of a given graph G with color lists $(L_v)_{v\in V(G)}$. We use this to conclude the lower bounds of Theorem 1 and Theorem 2.

4.1 Controlling the number of extensions modulo p

Our main gadget can be attached to a given set of vertices, and has the property that for each precoloring of the “glued on” vertices, there is a specified number of extensions. This is made precise in the result below.

▶ **Theorem 14.** *Let $k \in \mathbb{N}$ and $f : [q]^k \to \mathbb{N}$. There exist a graph G_f, a set of vertices $B = \{b_1, \dots, b_k\} \subseteq V(G_f)$ of size k and lists $(L_v)_{v\in V(G_f)}$, such that for any $\alpha \in [q]^k$, there are exactly $f(\alpha)$ list q-colorings c of G_f with $c(b_i) = \alpha(i)$ for all $i \in [k]$. Additionally, $|V(G_f)| \leq 20kq^{k+1}\max(f)$ and G_f has cutwidth at most $6kq^{k+2}$.*

The proof is given in Appendix B.

4.2 Reduction for counting q-colorings modulo p

In this section we prove the following result.

▶ **Theorem 15.** *Let p be a prime and let $q \in \mathbb{N}$ such that p does not divide $q-1$. Assuming SETH, there is no $\varepsilon > 0$ for which there exists an algorithm that counts the number of list q-colorings modulo p for a given n-vertex graph, with a given cut decomposition of width* ctw, *in time $(q-\varepsilon)^{\mathrm{ctw}} n^{O(1)}$.*

Suppose now that p divides $q-1$. Let $q' = q-1$. Then p does not divide $q'-1 = q-2$ and so the result above applies. Noting that any algorithm for #List q-Coloring also works for #List q'-Coloring, we find the following corollary.

▶ **Corollary 16.** *Let p be a prime and let $q \in \mathbb{N}$ such that p divides $q-1$. Assuming SETH, there is no $\varepsilon > 0$ for which there exists an algorithm that counts the number of list q-colorings modulo p for a given n-vertex graph, with a given cut decomposition of width* ctw, *in time $(q-1-\varepsilon)^{\mathrm{ctw}} n^{O(1)}$.*

Combining the two results above with Theorem 8 gives Theorem 1.

We use the notion of *constraint satisfaction problems* (CSP). Informally, a CSP asks if there is an assignment of values from a given domain to a set of variables such that they satisfy a given set of relations. We denote by $\mathrm{CSP}(q,r)$ the CSP with domain $[q]$ and constraints of arity at most r. We use $\#\mathrm{CSP}(q,r)$ to denote the problem of counting the number of solutions of a given instance of $\mathrm{CSP}(q,r)$. We use the following result from [12].

▶ **Theorem 17** ([12], Theorem 2.5)**.** *For each prime p, for every integer $q \geq 2$ and $\varepsilon > 0$ there is an integer r, such that the following holds. Unless the SETH fails, $\#_p CSP(q,r)$ with n variables and m constraints cannot be solved in time $(q-\varepsilon)^n (n+m)^{O(1)}$.*

This theorem follows from the proof of [12, Theorem 2.5], since their reduction preserves the number of solutions.

Proof of Theorem 15. Let $q \in \mathbb{N}$ and let p be a prime that does not divide $q-1$. Fix $\epsilon > 0$ and let r be given from Theorem 17. We will reduce a given instance of $\#_p\mathrm{CSP}(q,r)$ with constraints $C_1, \dots, C_m$ and variables $x_1, \dots, x_n$ to an instance (G, L) of $\#_p$List q-Coloring on $O_{p,r,q}(nm)$ vertices of cutwidth $n + O_{p,r,q}(1)$.

The graph G contains $2m$ columns with n vertices: for each constraint C_j, and for each variable x_i, we create two vertices $s_{i,j}$ and $t_{i,j}$ (where $j \in [m]$ and $i \in [n]$), which all get $\{1, \dots, q\}$ as list. For all $j \in [m-1]$, we place an edge between $t_{i,j}$ and $s_{i,j+1}$.

The color assigned to $s_{i,1}$ will be interpreted as the value assigned to variable x_i. Fix $j \in [m]$. We create gadgets on some vertex set V_j using Theorem 14, that are "glued" on subsets of vertices from $C_j = \{s_{i,j}, t_{i,j} : i \in [n]\}$.

1. For each $i \in [n]$, if $j < m$, we create a gadget on boundary set $\{s_{i,j}, t_{i,j}\}$ which ensures that we may restrict to counting list colorings c of (G, L) with $c(s_{i,j}) = c(s_{i,j+1})$.
2. There is a gadget on a boundary set of size at most r (the $s_{i,j}$ corresponding to the variables involved in the jth constraint), for which the number of extensions of any coloring of the boundary to this gadget is equivalent to 0 modulo p whenever the jth constraint is not satisfied, and equal to one otherwise.

A broad overview of the construction is depicted in Figure 2.

For the first property, we need the fact that p does not divide $q-1$: this ensures that the color compatibility matrix of a single edge is invertible, which will allow us to "transfer all information about the colors". The precise construction of the gadgets is deferred to Appendix B.

We obtain a cutwidth decomposition of the graph by first running over the vertices in the order

$$C_1 \cup V_1,\ C_2 \cup V_2, \dots, C_m \cup V_m.$$

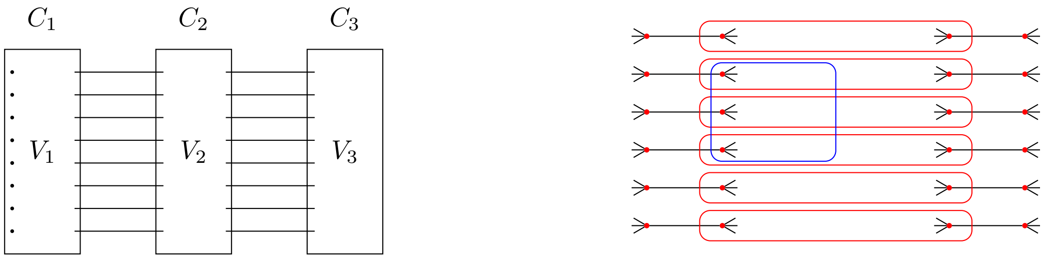


Figure 2 A sketch overview of the construction is given on the left-hand side and a more detailed view of two of the columns is given on the right-hand side. The red areas ensure the preservation of information, as described in point 1. The blue area checks whether the clause is satisfied, as described in point 2.

Within $C_j \cup V_j$, we first list $s_{1,j}, t_{1,j}$ and the vertices in the gadget that has those vertices as boundary set, and then repeat this for $s_{2,j}, t_{2,j}$, etcetera. Finally, we run over the vertices in the gadget that verifies the jth constraint. At each point, the cutwidth is bounded by n plus a constant (that may depend on p, q and r). ◀

4.3 Corollaries

We now extend the lower bound of Theorem 15 to counting connected edge sets via the following problem.

▶ **Definition 18.** *Given a graph G, two q-colorings c and c' are equivalent if there is some permutation $\pi : [q] \to [q]$ such that $c = \pi \circ c'$. We will refer to these equivalence classes as* essentially distinct q-colorings *and denote the problem of counting the number of essentially distinct q-colorings modulo a prime p by $\#_p$*ESSENTIALLY DISTINCT *q*-COLORING.

A simple reduction now gives us the following lower bound for $\#_p$ESSENTIALLY DISTINCT q-COLORING.

▶ **Corollary 19.** *Let p be a prime and $q \in \mathbb{N}$ an integer such that p does not divide $q-1$. Assuming SETH, there is no $\epsilon > 0$ for which there exists an algorithm that counts the number of essentially distinct q-colorings mod p for a given n-vertex graph that is not $(q-1)$-colorable, with a given cut decomposition of cutwidth* ctw*, in time $(q-\epsilon)^{\mathrm{ctw}} n^{O(1)}$.*

Proof. Let (G, L) be an instance of list coloring with cut decomposition $v_1, \ldots, v_n$. We construct an instance of $\#_p$ESSENTIALLY DISTINCT q-COLORING. The graph G' has vertex set

$$V(G') = V(G) \cup \{u^i_c : c \in [q], i \in [n]\}.$$

We add edges such that the vertices $\{u^i_c : c \in [q]\}$ induce a q-clique for all $i \in [n]$, and for $i \in [n-1]$ we add the edges $u^i_c u^{i+1}_{c'}$ for all $c \neq c'$. This ensures that, if u^1_c is colored c, then u^i_c is colored c for all $i \in [n]$. We now also add edges $u^i_c u_i$ for all $c \notin L_{v_i}$. Our new cut decomposition is

$$v_1, u^1_1, \ldots, u^1_q, v_2, u^2_1, \ldots, u^{n-1}_q, v_n, u^n_1, \ldots, u^n_q.$$

Note that $\mathrm{ctw}(G') \leq \mathrm{ctw}(G) + q^2$, $|V(G')| \leq (q+1)|V(G)|$ and that G' is not $(q-1)$-colorable. By Theorem 1, it suffices to show that the number of essentially distinct q-colorings of G' equals the number of list q-colorings of (G, L). We will do this by defining a bijective map.

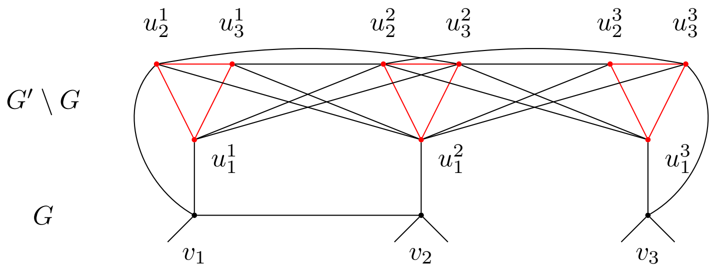


Figure 3 Example of the construction on (a part of) a graph G, with the cliques indicated in red. In this case $q = 3$ and we have $L_{v_1} = \{3\}, L_{v_2} = \{2, 3\}$ and $L_{v_3} = \{2\}$.

Let α be a list coloring of (G, L). Then we can color G' by setting $\alpha'(v) = \alpha(v)$ for $v \in V(G)$ and $\alpha'(u_c^i) = c$ for $c \in [q]$ and $i \in [n]$. This gives us a mapping $\gamma : \alpha \mapsto \overline{\alpha'}$, where $\overline{\alpha'}$ is the equivalence class of α'. We find an inverse map by first fixing a representative α' for $\overline{\alpha'}$, such that $\alpha'(u_c^1) = c$ for $c \in [q]$. We can do this since $G'[\{u_1^1, \dots, u_q^1\}]$ is a clique and thus each u_c^1 must get a unique color. Also note that since every color is now used, the rest of the coloring is also fixed and thus we find a unique representative this way. We now map $\overline{c'}$ to $c'|_{V(G)}$. Note that these two maps are well defined and compose to the identity map. We conclude that the number of list colorings of (G, L) is equal to the number of essentially distinct colorings of G'. ◀

To achieve the lower bound in Theorem 2, we use an existing argument from [1] to extend this bound to $\#_p$Connected Edge Sets. For this we will need the following definition.

▶ **Definition 20.** *The k-*stretch *of a graph G is the graph obtained from G by replacing each edge with a path of length k. We denote the k-stretch of G by ${}^kG$.*

Note that ${}^kG$ has the same cutwidth as G.

We now show that, assuming SETH, there is no $\epsilon > 0$ for which there exists an algorithm that counts the number of spanning connected edge sets mod p of n-vertex graphs of cutwidth at most ctw in time $O((p - \epsilon)^{\text{ctw}} n^{O(1)})$.

Proof. This proof closely follows a reduction from Annan [1], using ideas from Jaeger, Vertigan and Welsh [17].

Let G be any graph with cutwidth ctw and let p be a prime. Note that the number of spanning connected edgesets of G is equal to the value of $T(G; 1, 2)$, the Tutte polynomial of G at the point $(1, 2)$. The following formula is found in ([17], proof of Theorem 2)

$$T({}^kG; a, b) = (1 + a + \cdots + a^{k-1})^{n-r(G)} T\left(G; a^k, \frac{b + a + \cdots + a^{k-1}}{1 + a + \cdots + a^{k-1}}\right).$$

Choosing $a = 1, b = 2$ and $k = p - 1$, gives

$$T({}^{p-1}G; 1, 2) = (p - 1)^{n-r(G)} T\left(G; 1, \frac{2 + p - 2}{p - 1}\right) \equiv_p (-1)^{n-r(G)} T(G; 1 - p, 0).$$

Here we use the fact that for any multivariate polynomial $P(x, y) \equiv_p P(x + tp, y + sp)$ for any $s, t \in \mathbb{Z}$. We find that, since the k-stretch of a graph G has the same cutwidth as G, an algorithm that counts the number of spanning connected edgesets (mod p) on a graph with bounded cutwidth, also gives the Tutte polynomial at $(1 - p, 0)$ mod p.

Using another well known interpretation of the Tutte polynomial [23] we can relate the $T(G;1-p,0)$ to the chromatic polynomial $P(G;p)$ as follows:

$$P(G;p) = (-1)^{r(G)}p^{k(G)}T(G;1-p,0).$$

Note that we may assume that the number of connected components $k(G) = 1$ (and thus $r(G) = n-1$), since the number of spanning connected edgesets is trivially 0 if G has more than one component. We want to get rid of the remaining factor of p (since we will work mod p). To do this we will count the number of essentially distinct colorings (different up to color-permutations) using exactly p colors instead. This will turn out to be at least as hard as counting all colorings.

Let G be a graph that is not $(p-1)$-colorable. With this assumption, the number of p colorings of G is $p!$ times the number $C_p(G)$ of essentially distinct p-colorings of G, since any coloring uses all colors and thus can be mapped to $p!$ equivalent colorings by permuting the colors. So

$$(-1)^{n-1}pT(G,1-p,0) = P(G;p) = p(p-1)!C_p(G),$$

which holds over the real numbers hence we may divide both sides by p. By Wilson's Theorem $(p-1)! \equiv_p -1$, so we find

$$(-1)^{n-1}T(G,1-p,0) \equiv_p -C_p(G).$$

Hence we can use the number of spanning connected edgesets (mod p) of the $(p-1)$-stretch of G to find the Tutte polynomial at $(1-p,0)$ mod p and then also the number of essentially distinct colorings. The lower bound of Theorem 2 now follows from Corollary 19 (with $q = p$). The upper bound is proved in the full version of our paper [15]. ◀

5 Conclusion

In this paper we give tight lower and upper bounds for counting the number of (list) q-colorings and connected spanning edge sets of graphs with a given cutwidth decomposition of small cutwidth. Our results specifically relate to list q-coloring and essentially distinct q-coloring, but they can easily be extended to normal q-coloring for certain cases. In particular, if $q < p$, we may apply Corollary 19, since in the setting of the corollary, the values differ by $q!$ which is nonzero modulo p. If the chromatic number $\chi(G) \geq p$, then the number of q-colorings is trivially $0 \mod p$, since the number of q-colorings is a multiple of $\chi(G)$. This leaves us with the rather specific case of $\chi(G) < p \leq q$, for which the exact complexity remains unresolved.

Our results on the modular counting of colorings show that the modulus can influence the complexity in interesting ways, and that in some cases this effect can be directly explained by the rank of the compatibility matrix.

Our results leave several directions for further research:

- What is the fine-grained complexity of evaluating other points of the Tutte polynomial (modulo p)?
- What is the complexity of counting homomorphisms to graphs different from complete graphs, e.g. cycles or paths. Is it still determined by the rank of an associated compatibility matrix?

Another question in the direction of fast *decision problems* is how small representative sets we can get for the compatibility matrix of graphs other than complete bipartite graphs (which is equivalent to the setting of [13, Theorem 1.2]) or matchings which has been studied for the decision version in [18].

References

1 James Douglas Annan. *The complexity of counting problems*. PhD thesis, University of Oxford, 1994.

2 Andreas Björklund and Thore Husfeldt. Inclusion-exclusion based algorithms for graph colouring. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 13. Citeseer, 2006.

3 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the tutte polynomial in vertex-exponential time. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 677–686. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.40.

4 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

5 Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. *Journal of Computer and System Sciences*, 74(3):386–393, 2008.

6 Radu Curticapean, Nathan Lindzey, and Jesper Nederlof. A tight lower bound for counting hamiltonian cycles via matrix rank. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1080–1099. SIAM, 2018. doi:10.1137/1.9781611975031.70.

7 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.ch113.

8 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

9 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Joham MM van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 150–159. IEEE, 2011.

10 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.

11 Martin Dyer and Catherine Greenhill. The complexity of counting graph homomorphisms. *Random Structures & Algorithms*, 17(3-4):260–289, 2000.

12 Jacob Focke, Dániel Marx, and Paweł Rzążewski. Counting list homomorphisms from graphs of bounded treewidth: tight complexity bounds. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 431–458. SIAM, 2022.

13 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

14 Catherine Greenhill. The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. *Computational Complexity*, 9(1):52–72, 2000.

15 Carla Groenland, Jesper Nederlof, Isja Mannens, and Krisztina Szilágyi. Tight bounds for counting colorings and connected edge sets parameterized by cutwidth. *arXiv preprint*, 2021. arXiv:2110.02730.

16 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

17 François Jaeger, Dirk L Vertigan, and Dominic JA Welsh. On the computational complexity of the Jones and Tutte polynomials. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 108(1), pages 35–53. Cambridge University Press, 1990.

18 Bart MP Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *Theoretical Computer Science*, 795:520–539, 2019.

19 Amirhossein Kazeminia and Andrei A Bulatov. Counting homomorphisms modulo a prime number. *arXiv preprint*, 2019. arXiv:1905.10682.

20 Mikko Koivisto. An $o^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 583–590. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.11.

21 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

22 Jesper Nederlof. Bipartite TSP in $O(1.9999^n)$ time, assuming quadratic time matrix multiplication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 40–53. ACM, 2020. doi:10.1145/3357713.3384264.

23 James G Oxley, Dominic JA Welsh, et al. The tutte polynomial and percolation. *Graph Theory and Related Topics*, pages 329–339, 1979.

24 Leslie G. Valiant. Accidental algorithms. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 509–517. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.7.

25 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

A Proof of Lemma 12

Let H a bipartite graph with parts X, Y and associated color compatibility matrix M. Let $T : \mathrm{col}(X) \to \mathbb{F}_p$ be a function with reduced vertices $R \subseteq X$ and let vertex $v \in X \setminus R$ be a vertex of degree 1 in H. We need to find a function $T' : \mathrm{col}(X) \to \mathbb{F}_p$ with reduced vertices $R \cup \{v\}$ that is M-representative of T. (And need to show this can be done in time $O((q-1)^{|R|}q^{|X|-|R|})$.)

We may restrict to colorings x that do not assign value q to any element of R. There are at most $(q-1)^{|R|}q^{|X \setminus R|}$ such colorings. We set

$$T'[x] = \begin{cases} 0, & \text{if } x(v) = q, \\ T[x] - T[x'], \text{ where } x' \text{ is obtained from } x \text{ by changing the value of } v \text{ to } q, & \text{otherwise.} \end{cases}$$

This computation is done in time linear in the number of the colorings x we consider, so the running time is in $O((q-1)^{|R|}q^{|X \setminus R|})$.

First we will show that T' is an M-representative of T. Let $y \in \mathrm{col}(Y)$ be a coloring of the right hand side of the bipartite graph H. We need to show that

$$\sum_{\substack{x \in \mathrm{col}(X) \\ x \sim y}} T[x] \equiv_p \sum_{\substack{x \in \mathrm{col}(X) \\ x \sim y}} T'[x].$$

By definition,

$$\sum_{\substack{x \in \mathrm{col}(X) \\ x \sim y}} T'[x] = \sum_{\substack{x \in \mathrm{col}(X) \\ x \sim y \\ x(v) = q}} 0 + \sum_{\substack{x \in \mathrm{col}(X) \\ x \sim y \\ x(v) \neq q}} T[x] - T[x'].$$

Thus it remains to show that

$$\sum_{\substack{x\in \mathrm{col}(X)\\ x\sim y\\ x(v)\neq q}} -T[x'] \equiv_p \sum_{\substack{x\in \mathrm{col}(X)\\ x\sim y\\ x(v)=q}} T[x].$$

Let $x \in \mathrm{col}(X)$ with $x(v) = q$. We show the equality by proving that the number of times $T[x]$ appears on the left hand side equals the number of times $T[x]$ appears on the right hand side, modulo p. Let $w \in Y$ be the unique neighbor of the vertex v.

First assume that $x \sim y$. Then $y(w) \neq q$. If we adjust x to the coloring x_i, which is equal to x apart from assigning color i to v instead of q, then $x_i \sim y$ if and only if $i \neq y(w)$. Hence the term $-T[x]$ appears $q-2$ times on the left hand side, and $T[x]$ appears once on the right hand side. Since p divides $q-1$, we find $q-2 \equiv_p -1$ and hence both contributions are equal modulo p.

If $x \not\sim y$, then either x does not appear on both sides (because $x|_{X\setminus\{v\}}$ is already incompatible with y) or $y(w) = q$. If $y(w) = q$, then the term $T[x]$ appears $q-1 \equiv_p 0$ times on the left hand side by a similar argument as the above, and does not appear on the right hand side. This shows the claimed equality and finishes the proof.

B Proofs omitted from Section 4

B.1 Proof of Theorem 14

We first reduce the lists of each b_i to $\{1,2\}$ using the following gadget.

▶ **Lemma 21.** *Let $q, k \in \mathbb{N}$ and let $a \in [q]$. There is a graph G with $b, b' \in V(G)$ and color lists $L_v \subseteq [q]$ for $v \in V(G)$, such that $L_b = [q]$ and the following two properties hold:*

- *for all $c_b \in [q]$, there is a unique list coloring c of G with $c(b) = c_b$,*
- *for all list colorings c of G, if $c(b) = a$, then $c(b') = 1$ and if $c(b) \neq a$ then $c(b') = 2$.*

Proof. We first note that it is easy to "relabel colors", as shown in the construction[3] in Figure 4. We can therefore first make a gadget for which b' has color list $\{a, a'\}$ for some $a' \neq a$, and then relabel a, a' to $1, 2$. By symmetry, we can therefore assume that $a = 1$ (or simply replace 1 with a and 2 with a' in the argument below). Let

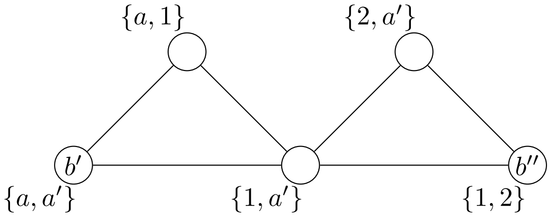


Figure 4 A gadget to "relabel colors". It has two special vertices b' and b'', and lists are depicted with sets. For any list coloring c of the depicted gadget, if $c(b') = a$, then $c(b'') = 1$ and if $c(b') = a'$, then $c(b'') = 2$. In both cases, there is a unique way to color the remaining vertices.

$$V = \{b, b'\} \cup \{s_i : i = 2, \dots, q\} \cup \{t_i : i = 2, \dots, q\}$$

and

[3] If $a = 1$ or $a' = 2$ we slightly change the construction by removing the top left or top right vertex respectively.

$$E = \{s_i b : i = 2, \dots, q\} \cup \{s_i b' : i = 2, \dots, q\} \cup \{s_i t_j : i, j = 2, \dots, q\}.$$

Now let $L_b = [q]$, $L_{b'} = \{1, 2\}$ and $L_{t_i} = L_{s_i} = \{1, i\}$ for $i \in \{2, \dots, q\}$. A depiction is given in Figure 5.

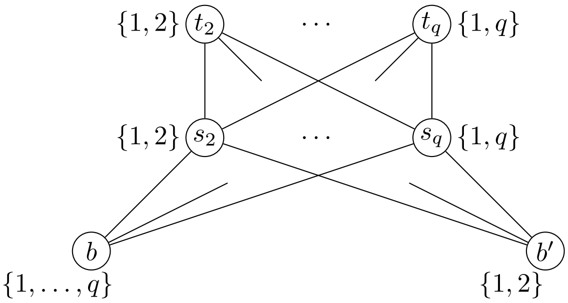


Figure 5 The construction of the list coloring instance of the proof of Lemma 21.

If a list coloring c of G satisfies $c(b) = 1$, then $c(s_i) = i$ and thus $c(t_i) = 1$ for each $i \in \{2, \dots, q\}$. In particular $c(s_2) = 2$ and $c(b') = 1$.

If $c_b \in \{2, \dots, q\}$, then any list coloring c with $c(b) = c_b$ satisfies $c(s_i) = 1$ and $c(t_i) = i$ for all $i \in \{2, \dots, q\}$, and so $c(b') = 2$.

This proves that, starting with the color $c_b \in [q]$ for b, there is always a unique extension to a list coloring of G, and this satisfies the property that vertex b' receives color 1 if $c_b = 1$, and receives color 2 otherwise. ◀

We also make use of the following construction.

▶ **Lemma 22.** *Let $k, \ell \in \mathbb{N}$. There is a graph G, a subset of vertices $B = \{b_1, \dots, b_k\} \subseteq V$ of size k, and color lists L_v for all $v \in V(G)$ such that:*

- *$L_{b_i} = \{1, 2\}$ for all $i \in \{1, \dots, k\}$,*
- *there are exactly ℓ list colorings c of G with $c(B) = \{1\}$,*
- *for each partial coloring c_B of B with $c_B(B) \neq \{1\}$, there is a unique extension of c_B to a list coloring of G.*

Proof. We start with $V = B$ and add a path[4] $w_1, \dots, w_{\ell-1}$ with color lists

$$L_{w_i} = \begin{cases} \{2,3\} & \text{if } i \equiv_3 1, \\ \{1,3\} & \text{if } i \equiv_3 2, \\ \{1,2\} & \text{if } i \equiv_3 0, \end{cases}$$

and add edges $b_i w_1$ for $i = 1, \dots, k$. A depiction is given in Figure 6.

If a list coloring c satisfies $c(b_i) = 2$ for some $i \in [k]$, then $c(w_1) = 3$, $c(w_2) = 1$, $c(w_3) = 2$ etcetera. Hence there is a unique extension of any partial coloring of B that assigns color 2 somewhere.

If $c(b_i) = 1$ for all $i \in [k]$, then we have a choice for the color of w_1. If $c(w_1) = 3$ then we get the same propagation as before, however if $c(w_1) = 2$, then we have a choice for the color of w_2. Using a simple induction argument we find that the number of possible list colorings with $c(B) = 1$ equals ℓ. ◀

[4] When $\ell = 1$, we add no vertices of the form w_i and the statements of the lemma immediately follow.

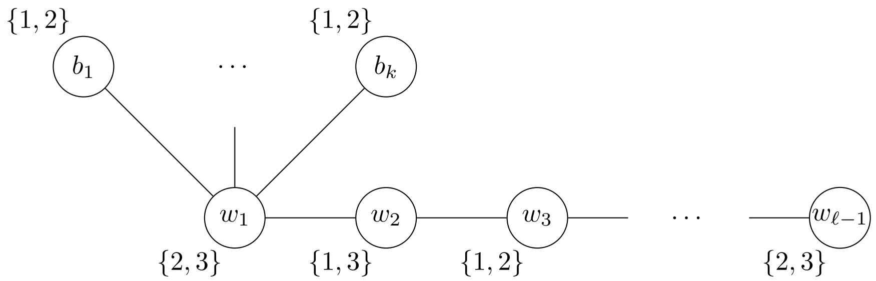


Figure 6 Construction in the proof of Lemma 22 when $\ell \equiv_3 2$.

We are now ready to construct the main gadget.

Proof of Theorem 14. Let $f \in [q]^k$. We create vertices $b_1, \ldots, b_k$ and give them all $[q]$ as list.

For each partial coloring $\alpha \in [q]^k$, we create a graph G_α that contains $b_1, \ldots, b_k$ in their vertex set, but the graphs are on disjoint vertex sets otherwise (we "glue" the graphs on the special vertices $b_1, \ldots, b_k$). It suffices to show that we can find lists for the "private" vertices of G_α such that the number of extensions of a coloring c_B of B is 1 if $c_B(b_i) \neq \alpha(i)$ for some $i \in [k]$, and $f(\alpha)$ otherwise. The resulting gadget will then have $1 \cdot 1 \cdot \ldots \cdot 1 \cdot f(\alpha) = f(\alpha)$ possible extensions for the precoloring α, as desired.

We now turn to constructing the gadget G_α for a fixed coloring $\alpha \in [q]^k$. We first reduce to the case in which each b_i has $\{1,2\}$ as list. Let $i \in [k]$. Using Lemma 21 with $a = \alpha(i)$, we obtain a gadget $H_{b,b'}$ and identify the special vertex b with b_i. For each α, we obtain a new set of vertices $b'_1, \ldots, b'_k$ with lists $\{1,2\}$. We then glue these onto the special vertices from a gadget obtained by applying Lemma 22 with $\ell = f(\alpha)$. If $b_1, \ldots, b_k$ are colored as specified by α, then $b'_1, \ldots, b'_k$ all receive color 1 and G_α has $f(\alpha)$ possible extensions; however if some b_i receives the wrong color, the corresponding b'_i receives color 2 and there is a unique extension to the rest of G_α.

It remains to show the bounds on the number of vertices and the cutwidth. We give the very rough upperbound of $6kq^{k+2}$ on the cutwidth. The gadget from Lemma 21 has cutwidth at most $q^2 + 6$ (since this is an upper bound on the number of edges in that construction). The gadgets from Lemma 22 have cutwidth at most k. A final cut decomposition can be obtained by first enumerating the vertices in B, and then adding the cut decompositions of each G_α, one after the other.

Finally the number of vertices of the graph is upper bounded by q^k times the maximum number of vertices of the graph G_α. The gadget of Lemma 21 has at most $2q + 6$ vertices and there are k of them, so they contribute at most $12kq$ vertices. The gadgets from Lemma 22 add at most $f(\alpha)$ vertices. In total,

$$|V(G_f)| \leq 20kq^{k+1} \max(f). \qquad \blacktriangleleft$$

B.2 Remaining details of the proof of Theorem 15

We first describe the gadgets for the "color transfer" (the first desired property). Let $j \in [m-1]$ and $i \in [n]$. We will apply Theorem 14 to a function $f_{i,j}$ with boundary set $B_{i,j} = (s_{i,j}, t_{i,j})$ and $\max(f_{i,j}) = p$, resulting in a graph on $O_{p,q}(1)$ vertices. Let J_1 be the $q \times q$ coloring compatibility matrix of a single edge, and let J_1^{-1} denote its inverse over $\mathbb{F}_p$

(that is, $J_1^{-1} J_1 \equiv_p I_q$, the $q \times q$ identity matrix). We "choose a representative" $\widetilde{J_1}$, which has entries in $\{1, \ldots, p\}$ that are equivalent to those in J_1^{-1} modulo p. For $c_1, c_2 \in [q]$ possible colors for $s_{i,j}$ and $t_{i,j}$ respectively, we set

$$f_{i,j}(c_1, c_2) = \widetilde{J_1}[c_1, c_2].$$

Let $V_{i,j}$ denote the vertices in the gadget obtained by applying Theorem 14 to $(f_{i,j}, B_{i,j})$ that are not in $B_{i,j}$. Let $c_1, c_3 \in [q]$. The number of list colorings c of the graph induced on $B_{i,j} \cup V_{i,j} \cup \{s_{i,j+1}\}$ with $c(s_{i,j}) = c_1$ and $c(s_{i,j+1}) = c_3$ is equal to

$$\sum_{c_2 \in [q]} f_{i,j}(c_1, c_2) J_1[c_2, c_3] = (\widetilde{J_1} J_1)[c_1, c_3],$$

since for any coloring c_2 we have $f_{i,j}(c_1, c_2) J_1[c_2, c_3]$ such colorings with $c(t_{i,j}) = c_2$, by definition of f_i, j and J_1. Therefore, modulo p this number of extensions is equal to 1 if $c_1 = c_3$ and 0 otherwise, as desired.

We now describe the gadgets that check the constraints. Let $j \in [m]$ and let $i_1, \ldots, i_\ell$ be given so that the jth constraint only depends on the variables $x_{i_1}, \ldots, x_{i_\ell}$ (where by assumption $\ell \leq r$). We will apply Theorem 14 to a function g_j with boundary set $B_j = (s_{i_1,j}, \ldots, s_{i_\ell,j})$ and $\max(g_j) = p$, resulting in a graph on $O_{p,q,\ell}(1) = O_{p,q,r}(1)$ vertices. We set $g_j(c_1, \ldots, c_\ell)$ to be equal to 1 if the assignment $(c_1, \ldots, c_\ell)$ to $(x_{i_1}, \ldots, x_{i_\ell})$ satisfies the jth constraint, and p otherwise. This ensures the second property described in the proof of Theorem 15.

Satisfiability of Circuits and Equations over Finite Malcev Algebras

Paweł M. Idziak
Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Piotr Kawałek
Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Jacek Krzaczkowski
Department of Computer Science, Faculty of Mathematics, Physics and Computer Science, Maria Curie-Sklodowska University, Lublin, Poland

Abstract

We show that the satisfiability of circuits over finite Malcev algebra $\mathbf{A}$ is NP-complete or $\mathbf{A}$ is nilpotent. This strengthens the result from our earlier paper [18] where nilpotency has been enforced, however with the use of a stronger assumption that no homomorphic image of $\mathbf{A}$ has NP-complete circuits satisfiability. Our methods are moreover strong enough to extend our result of [14] from groups to Malcev algebras. Namely we show that tractability of checking if an equation over such an algebra $\mathbf{A}$ has a solution enforces its nice structure: $\mathbf{A}$ must have a nilpotent congruence ν such that also the quotient algebra $\mathbf{A}/\nu$ is nilpotent. Otherwise, if $\mathbf{A}$ has no such congruence ν then the Exponential Time Hypothesis yields a quasipolynomial lower bound. Both our results contain important steps towards a full characterization of finite algebras with tractable circuit satisfiability as well as equation satisfiability.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Complexity classes

Keywords and phrases Circuit satisfiability, solving equations, Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.37

Funding The project is partially supported by Polish NCN Grant # 2014/14/A/ST6/00138.
Piotr Kawałek: This research is partially supported by the Priority Research Area SciMat under the program Excellence Initiative – Research University at the Jagiellonian University in Kraków.

1 Introduction

The problem of deciding whether an equation over an algebraic structure has a solution has got quite deep attention both in mathematics and computer science. Let us only mention two crucial examples:

- 10^{th} Hilbert's problem for Diophantine equations, i.e. equations over the ring of integers – shown to be undecidable by Matiyasevich [24],
- the problem SAT of satisfiability of Boolean formulas – shown to be NP-complete by Cook [4].

In this paper we consider equations of the form $\mathbf{t}(x_1,\ldots,x_n)=\mathbf{s}(x_1,\ldots,x_n)$, where $\mathbf{t}$ and $\mathbf{s}$ are polynomials over a fixed finite algebra $\mathbf{A}$ (i.e. a finite set A with finitely many operations), i.e. terms with some of their variables already evaluated by elements of $\mathbf{A}$. We are interested in the complexity of the problem PolSat($\mathbf{A}$), i.e. the problem of deciding whether an equation (of two polynomials) over $\mathbf{A}$ has a solution in $\mathbf{A}$.

Recently, this problem for equations over finite groups and other finite algebraic structures (like e.g. rings [10], semigroups [2, 21] or lattices [25]) attracted many researchers. For groups the story began with the paper [7] of Goldmann and Russell where NP-completeness of PolSat has been shown for nonsolvable groups and a polynomial time algorithm has been created for nilpotent groups. However the gap between solvable and nilpotent groups remained unfilled. More recently several examples of solvable but non-nilpotent groups with PolSat tractable in polynomial time have been provided [12, 13, 11, 5]. Among such groups there is the symmetric group $\mathbf{S}_3$ and the alternating group $\mathbf{A}_4$. Those two examples are of a special interest, as after endowing them by additional operations definable in terms of group multiplication the PolSat problem becomes NP-complete for such extensions. For the group $\mathbf{S}_3$ this phenomenon has been first described in [8], while for $\mathbf{A}_4$ in [13]. The NP-hardness for $\mathbf{A}_4$ has been obtained by extending this group by the binary commutator operation $[x, y] = x^{-1}y^{-1}xy$, which is heavily used in group theory.

The existence of such examples made the expectation of characterizing finite algebras with PolSat solvable in polynomial time rather hopeless. For this reason our paper [18] modified PolSat to make it independent of the choice of the basic operations in the algebra. This has been done by interpreting the size of a term (or a polynomial) not as the size of the tree that represent this term but as the size of a circuit computing it. One can easily see that the tree (built up with group multiplication only) computing the n-ary term $[\ldots[[x_1, x_2], x_3]\ldots x_n]$ has exponential size, while there is a circuit of linear size computing this term. This small change allows us to expand a finite algebra $\mathbf{A}$ by (finitely many) operations definable by polynomials of $\mathbf{A}$ without actually changing the complexity. Thus by a circuit satisfiability over an algebra $\mathbf{A}$ we mean the following problem

CSat($\mathbf{A}$): given a circuit over $\mathbf{A}$ with two output gates $\mathbf{g}_1, \mathbf{g}_2$, is there an assignment of values to the input gates $\overline{x} = (x_1, \ldots, x_n)$ that gives the same output on $\mathbf{g}_1, \mathbf{g}_2$, i.e. $\mathbf{g}_1(\overline{x}) = \mathbf{g}_2(\overline{x})$.

Note here that the characterizations given in [10, 25] show that the problems PolSat and CSat are tractable for the same rings and lattices: namely the only tractable rings are the nilpotent rings and the only tractable lattices are the distributive lattices.

The paper [18] shows that replacing polynomials by circuits representing those polynomials allows us to attack the complexity of CSat in a more general setting than just for particular algebraic structures like groups, rings or lattices. The setting considered there includes all of the above structures and many more, i.e. algebras from the so-called congruence modular varieties. Roughly speaking, most of the structures in classical abstract algebra (except semigroups) are included.

In this paper we improve the result of [18] and generalize the result of [14] from groups to more general algebraic structures. First, in both of those two improvements we restrict ourselves to the so-called Malcev algebras, i.e. algebras having a ternary term $\mathbf{d}(x, y, z)$ that satisfies $\mathbf{d}(x, x, y) = y = \mathbf{d}(y, x, x)$. Note that groups and in fact all algebras that are extensions of groups (like rings or Boolean algebras) are Malcev, as the term $\mathbf{d}(x, y, z) = xy^{-1}z$ does the job. Also many generalizations of groups, like quasigroups or loops, are Malcev. Next we refer to the monograph [6] where a commutator theory is developed in a way that generalizes the commutator $[H, K]$ of normal subgroups H, K in the group theory and the ideal multiplication $I \cdot J$ in the ring theory. In general setting we use congruences instead of normal subgroups or ideals. The book [6] shows how for two congruences α, β of an algebra $\mathbf{A}$ define their commutator $[\alpha, \beta]$ and can serve as a reference source. With the help of the commutator one can define notions of abelianess, solvability and nilpotency for arbitrary algebras. First, for a congruence θ and $i = 1, 2, \ldots$ we put

$$\begin{array}{rclcrcl} \theta^{(0)} & = & \theta & \qquad & \theta^{[0]} & = & \theta \\ \theta^{(i+1)} & = & [\theta, \theta^{(i)}] & & \theta^{[i+1]} & = & [\theta^{[i]}, \theta^{[i]}]. \end{array}$$

Now, a congruence θ of $\mathbf{A}$ is called *k-nilpotent* [or *k-solvable*] if $\theta^{(k)} = 0_{\mathbf{A}}$ [$\theta^{[k]} = 0_{\mathbf{A}}$] and the algebra $\mathbf{A}$ is *nilpotent* [*solvable*] if 1_A is k-nilpotent [k-solvable] for some finite k. In particular θ [or $\mathbf{A}$] is Abelian if $[\theta, \theta] = 0_{\mathbf{A}}$ [or $[1_{\mathbf{A}}, 1_{\mathbf{A}}] = 0_{\mathbf{A}}$]. In a similar way we can define what it means for a congruence θ to be Abelian, nilpotent or solvable over a smaller congruence α, by simply saying that an appropriate commutator power of θ is contained in α. Finally we define $\theta^{(\omega)} = \bigcap_{i=0}^{\infty} \theta^{(i)}$, and note that since in a finite algebra the descending chain $\theta^{(0)} \geqslant \theta^{(1)} \geqslant \theta^{(2)} \geqslant \ldots$ stabilizes the congruence $\theta^{(\omega)}$ is in fact one of the $\theta^{(i)}$'s.

In our study of solvable Malcev algebra a series of congruences $0 = \nu_0 \leqslant \nu_1 \leqslant \ldots \leqslant \nu_{h-1} \leqslant \nu_h = 1_{\mathbf{A}}$ in which ν_i is the largest nilpotent congruence over ν_{i-1} will play a crucial role. This series is called the nilpotent series (or sometimes the Fitting series in the group theory), and we will use this teminology as well. We define the nilpotent (or Fitting) rank $\mathsf{nr}(\alpha)$ of a congruence α to be the smallest integer k for which there is a sequence of congruences $0 = \alpha_0 \leqslant \alpha_1 \leqslant \ldots \leqslant \alpha_{k-1} \leqslant \alpha_k = \alpha$ where each α_i is nilpotent over α_{i-1}. Note here that ν_k is the largest congruence with nilpotent rank k, so that we have $\mathsf{nr}(\alpha) \leqslant k$ iff $\alpha \leqslant \nu_k$. By the same token any solvable congruence in a Malcev algebra has finite nilpotent rank.

For a finite Malcev algebra $\mathbf{A}$ and a covering pair $\alpha \prec \beta$ of congruences (i.e. without any congruence between them) there are tools to describe the behaviour of $\mathbf{A}$ locally, depending on whether β is Abelian over α. In the case it is not, tame congruence theory (as described in [9]) tells us that $\mathbf{A}$ has a unary idempotent polynomial $\mathbf{e}$ (i.e. $\mathbf{e}(\mathbf{e}(x)) = \mathbf{e}(x)$ for all $x \in A$) with a two element range $\{0, 1\} = \mathbf{e}(A)$ so that the induced algebra $\mathbf{A}|_{\{0,1\}}$ (i.e. the set $\{0, 1\}$ with all the polynomials of $\mathbf{A}$ that preserve that set) has Boolean operations $\wedge, \vee, \neg$ definable by polynomials. As one can expect the presence of a Boolean behaviour results in NP-hardness of CSAT($\mathbf{A}$) (see the proof of Corollary 1.3 for details). If there is no local Boolean behaviour in $\mathbf{A}$, i.e. $\mathbf{A}$ behaves locally in the Abelian fashion, then $\mathbf{A}$ is solvable. Thus our goal is to understand solvable algebras with tractable POLSAT or even CSAT.

In [18] it was shown that if a finite Malcev algebra $\mathbf{A}$ is not nilpotent then $\mathbf{A}$ has a nonnilpotent quotient $\mathbf{A}'$ with NP-complete CSAT($\mathbf{A}'$). Here we significantly improve that result to be read as follows.

▶ **Theorem 1.1.** *If a finite Malcev algebra $\mathbf{A}$ is not nilpotent then CSAT($\mathbf{A}$) is NP-complete.*

Unfortunately [18] does not provide a proof that nilpotency is already strong enough to force tractability. In fact [16] describes examples of nilpotent Malcev algebras with CSAT outside P under the assumption of Exponential Time Hypothesis. On the other hand [18] (and independently [22]) provides an argument that supernilpotent Malcev algebras have tractable CSAT. Due to [20], for algebras with finitely many basic operations this stronger condition of supernilpotency simply means that an algebra is nilpotent and decomposes into a direct product of algebras of prime power order. (Note here that due to Sylow's results every nilpotent group is already supernilpotent; the same is true for rings). Obviously the examples from [16] are not supernilpotent. In fact they are rather far from being supernilpotent. The very same paper [16] isolates a concept of supernilpotent rank, similar to the nilpotent rank, with the help of supernilpotent congruences instead of nilpotent ones. Thus we say that the supernilpotent rank of a congruence α is at most k (and write $\mathsf{sr}(\alpha) \leqslant k$) if there is a sequence of congruences $0 = \alpha_0 \leqslant \alpha_1 \leqslant \ldots \leqslant \alpha_{k-1} \leqslant \alpha_k = \alpha$ in which each α_i is supernilpotent over α_{i-1}. (Note here that in [16] a congruence α with $\mathsf{sr}(\alpha) \leqslant k$ has been called k-step supernilpotent.) Analogously as in the case of the nilpotent series $(\nu_k)_k$ we can define the supernilpotent series $(\sigma_k)_k$ in which σ_k is the largest congruence of supernilpotent rank k (all that is needed to do that is to observe that the join of two supernilpotent congruences is supernilpotent). We also have $\mathsf{sr}(\alpha) \leqslant k$ iff $\alpha \leqslant \sigma_k$. Moreover, since every supernilpotent congruence is nilpotent, $\sigma_k \leqslant \nu_k$ and $\mathsf{nr}(\alpha) \leqslant \mathsf{sr}(\alpha)$. Note that the concepts of nilpotency and supernilpotency coincide in groups so that $\mathsf{nr}(\alpha) = \mathsf{sr}(\alpha)$ for every congruence α of a finite group. This however is not the case in general, as for nilpotent but not supernilpotent algebra $\mathbf{A}$ we have $\mathsf{nr}(\mathbf{A}) = 1 < \mathsf{sr}(\mathbf{A})$.

We have already noticed that the examples from [16] are not supernilpotent. Actually their supernilpotent rank is at least 3. Very recently Kompatscher [23] applied the technique from [16] to show that (under ETH) no finite nilpotent algebra with supernilpotent rank at least 3 has tractable CSat (but note here that Kompatscher use the term Fitting rank, for what we call here supernilpotent rank). However even $\mathsf{sr}(\mathbf{A}) \leqslant 2$ does not suffice to have tractable CSat (or ETH fails). Appropriate examples are created in [17].

The sequence of papers [16, 26, 14] explores the same idea to show that for a solvable group $\mathbf{G}$ with $\mathsf{nr}(\mathbf{G}) \geqslant 3$ the problem PolSat($\mathbf{G}$) is not tractable (if ETH holds). Actually [14] combines some premature results from [16] and [26]. Here we leave the group realm and use tame congruence theory (instead of well understood group techniques) to bound the nilpotent rank of solvable algebras with tractable PolSat.

▶ **Theorem 1.2.** *Let $\mathbf{A}$ be a finite solvable Malcev algebra with nilpotent rank $h \geqslant 3$. Then checking if an equation of length ℓ over $\mathbf{A}$ has a solution needs at least $2^{\Omega(\log^{h-1} \ell)}$ steps, or the Exponential Time Hypothesis fails.*

Combining Theorem 1.2 with the possibility of eliminating nonabelian (and therefore Boolean) local behaviour we will also get the following Corollary.

▶ **Corollary 1.3.** *Let $\mathbf{A}$ be a finite Malcev algebra. If PolSat($\mathbf{A}$) $\in$ P then $\mathbf{A}$ is solvable and $\mathsf{nr}(\mathbf{A}) \leqslant 2$, or ETH fails.*

We conclude the description of our results by noting that (under the Exponential Time Hypothesis) nilpotent rank 2 does not put PolSat into P, as some dihedral groups described in [17] show.

2 Proof of the Theorems

To prove the results formulated in the Introduction we need some preparation stated in two Lemmas below. Their proofs are postponed to Section 3 as they make some (or sometimes even quite heavy) use of the theory of commutator in congruence modular varieties (or the modular commutator theory, for short) and the tame congruence theory described in [6] and [9], respectively. This section however does not require the knowledge of these theories. All we need to state our Lemmas is the concept of a join irreducible congruence i.e. a congruence θ that cannot be represented as a join $\theta_1 \vee \theta_2$ for $\theta_1, \theta_2 < \theta$. Such a congruence has a unique subcover, which will be denoted by θ_-. Moreover θ has to be principal, i.e. it is generated by a single pair, say (a, b). This last fact is to be denoted by $\theta = \Theta(a, b)$. For these and all other basic algebraic concepts and notation we refer the reader to [3].

▶ **Lemma 2.1.** *Let* $\mathbf{A}$ *be a finite solvable Malcev algebra. Then for every join irreducible congruence* $\delta = \Theta(e, a)$ *with* $\mathsf{nr}\,(\delta) > \mathsf{nr}\,(\delta_-) > 0$ *there is another join irreducible congruence* $\delta^\star = \Theta(e^\star, a^\star)$ *with* $\mathsf{nr}\,(\delta^\star) = \mathsf{nr}\,(\delta) - 1$ *and* $\mathsf{nr}\,(\delta^\star) > \mathsf{nr}\left(\delta^\star_-\right)$*, and for each integer* n *there is an* n*-ary polynomial* $\mathbf{and}_n(x_1, \ldots, x_n)$ *such that for* $x_1, \ldots, x_n \in \{e, a\}$ *we have*

$$\mathbf{and}_n(x_1, \ldots, x_n) = \begin{cases} a^\star, & \text{if } x_1 = \ldots = x_n = a, \\ e^\star, & \text{otherwise.} \end{cases}$$

The polynomial $\mathbf{and}_n$ *can be constructed in a time bounded by* $2^{O(n)}$ *while the circuit that computes* $\mathbf{and}_n$ *can be constructed in a linear time* $O(n)$*.*

▶ **Lemma 2.2.** *In a finite solvable Malcev algebra* $\mathbf{A}$ *with nilpotent rank* $h \geqslant 2$ *there are:*

- *a join irreducible congruence* $\delta^{h-1} = \Theta(e_{h-1}, a_{h-1})$ *with* $h - 1 = \mathsf{nr}\left(\delta^{h-1}\right) > \mathsf{nr}\left(\delta^{h-1}_-\right) = h - 2$*,*
- *a partition of* A *into two nonempty disjoint subsets* $A = A_\perp \cup A_\top$*,*

such that for any 3*-CNF-formula* Φ *with* m *clauses there exists a* $3m$*-ary polynomial* $\mathbf{sat}_\Phi$ *of* $\mathbf{A}$ *with range contained in* $\{e_{h-1}, a_{h-1}\}$ *and such that for* $z^1_1, z^1_2, z^1_3, \ldots, z^m_1, z^m_2, z^m_3 \in \{\top, \perp\}$ *and* $x^1_1, x^1_2, x^1_3, \ldots, x^m_1, x^m_2, x^m_3 \in A$ *with* $x^j_i \in A_{z^j_i}$ *we have*

$$\Phi(z^1_1, z^1_2, z^1_3, \ldots, z^m_1, z^m_2, z^m_3) = \top \quad \textit{iff} \quad \mathbf{sat}_\Phi(x^1_1, x^1_2, x^1_3, \ldots, x^m_1, x^m_2, x^m_3) = a_{h-1}.$$

The polynomial $\mathbf{sat}_\Phi$ *can be constructed (from the formula* Φ*) in time bounded by* $2^{O(m)}$ *while the circuit that computes* $\mathbf{sat}_\Phi$ *can be constructed in linear time* $O(m)$*.*

Now we are ready to prove our results stated in the Introduction.

Proof of Theorem 1.2. We are going to translate a 3-CNF formula Φ with m clauses into an equation of length $2^{O(m^{1/(h-1)})}$ (and with the very same time needed for this translation) such that the formula Φ is satisfiable iff the corresponding equation has a solution. This, according to ETH (together with the Sparsification Lemma) shows that the time needed to check if an equation of length ℓ has a solution is at least $2^{\Omega(\log^{h-1} \ell)}$. For if not, a PolSat($\mathbf{A}$) algorithm working in $2^{o(\log^{h-1} \ell)}$ time would solve 3-CNF-SAT in $2^{o(m)}$, contrary to ETH.

Without loss of generality we assume that $m = k^{h-1}$. We will produce a $3m$-ary polynomial $\mathbf{\Phi Sat}$ represented by a tree having:

- exactly h levels,
- $3m$ leaves, all of them on the level h,
- $m/k = k^{h-2}$ nodes on level $h - 1$, each of which labeled by $3k$-ary polynomial of the form $\mathbf{sat}$ provided by Lemma 2.2,
- $m/k^{h-l} = k^{l-1}$ nodes at the l-th level (for $l = h - 2, ..., 1$), each of which labeled by k-ary polynomial of the form $\mathbf{and}_k$ supplied by Lemma 2.1.

To do that we start with a 3-CNF formula Φ with $m = k^{h-1}$ clauses and group them into m/k groups each of which containing exactly k clauses. Thus Φ can be represented as $\bigwedge_{j=1}^{m/k} \Phi_j$, with Φ_j being a conjunction of k clauses in the j-th group. Now, with the help of Lemma 2.2 we produce:

- the partition $A = A_\top \cup A_\perp$,
- the join irreducible congruence δ^{h-1} with $\mathsf{nr}\left(\delta^{h-1}\right) = h - 1 = 1 + \mathsf{nr}\left(\delta^{h-1}_-\right)$,
- the elements e_{h-1}, a_{h-1} with $\delta^{h-1} = \Theta(e_{h-1}, a_{h-1})$,
- and for each Φ_j a corresponding $3k$-ary polynomial $\mathbf{sat}_{\Phi_j}$ with the property described by the Lemma 2.2.

Next we go down with $l = h - 1, ..., 2$ to use Lemma 2.1 and for $\delta^l = \Theta(e_l, a_l)$ satisfying $\mathsf{nr}\left(\delta^l\right) = l$ we produce

- the congruence $\delta^{l-1} = \Theta(e_{l-1}, a_{l-1})$ with $\mathsf{nr}\left(\delta^{l-1}\right) = l-1$ by putting $\delta^{l-1} = \left(\delta^{l}\right)^{*}$, $e_{l-1} = e_l^*$ and $a_{l-1} = a_l^*$,
- the k-ary polynomial $\mathbf{and}_k^{l-1}$ with the range $\{e_{l-1}, a_{l-1}\}$, so that

$$\mathbf{and}_k^{l-1}(x_1,\dots,x_k) = \begin{cases} a_{l-1}, & \text{if } x_1 = \ldots = x_k = a_l, \\ e_{l-1}, & \text{otherwise,} \end{cases}$$

whenever $x_1,\dots,x_k \in \{e_l, a_l\}$.

To get the polynomial $\Phi\mathbf{Sat}$ we first compute $\mathbf{sat}_{\Phi_1}(\overline{x}),\dots,\mathbf{sat}_{\Phi_{m/k}}(\overline{x})$. Note that in fact in each of the $\mathbf{sat}_{\Phi_j}(\overline{x})$'s at most $3k$ variables (from $\overline{x}$) may occur, as the z_i^j's in different clauses do not have to be different. Next, to pass from level $l = h-1,\dots,2$ to $l-1$ we group k^{l-1} values into k^{l-2} groups, each of which having k elements and apply the k-ary polynomial $\mathbf{and}_k^{l-1}$ to each of these groups to get k^{l-2} values on level $l-1$. Note that, due the properties of the ranges of the $\mathbf{sat}_{\Phi_j}$'s and of the $\mathbf{and}_k^{l}$'s, the only values that may occur at the l-th level (with $l = h-1,\dots,1$) are e_l and a_l. Moreover the value a_l occurs only if the conjunction of k^{h-l} clauses that were used to compute this value is properly evaluated (to be satisfied). In particular arriving at level 1 we get one of the values e_1 or a_1 so that the resulting $3m$-ary polynomial $\Phi\mathbf{Sat}(x_1^1, x_2^1, x_3^1, \dots, x_1^m, x_2^m, x_3^m)$ satisfies

$$\Phi(z_1^1, z_2^1, z_3^1, \dots, z_1^m, z_2^m, z_3^m) = \top \quad \text{iff} \quad \Phi\mathbf{Sat}(x_1^1, x_2^1, x_3^1, \dots, x_1^m, x_2^m, x_3^m) = a_1,$$

whenever $z_1^1, z_2^1, z_3^1, \dots, z_1^m, z_2^m, z_3^m \in \{\top, \bot\}$ and $x_1^1, x_2^1, x_3^1, \dots, x_1^m, x_2^m, x_3^m \in A$ are such that $x_i^j \in A_{z_i^j}$. This means that the equation $\Phi\mathbf{Sat}(x_1^1, x_2^1, x_3^1, \dots, x_1^m, x_2^m, x_3^m) = a_1$ has a solution iff the formula Φ is satisfiable.

To conclude the proof we observe that Lemmas 2.1 and 2.2 guarantee that the polynomials of the form $\mathbf{sat}_{\Phi_j}$ and $\mathbf{and}_k^l$ have their size bounded by $2^{O(k)}$ and in fact they can be constructed in the very same amount of steps. Thus composing them to get $\Phi\mathbf{Sat}$ we need roughly $\left(2^{O(k)}\right)^h = 2^{O(m^{1/(h-1)})}$ steps, as promised. ◀

Proof of Corollary 1.3. As we have already mentioned in the Introduction a finite Malcev algebra admits only two kinds of a local behaviour. One of them is Boolean (or type **3** in the sense of tame congruence theory [9]). Modifying our argument used in Section 5 of [18] we show that the presence of type **3** leads to NP-completeness. Indeed, in this case the algebra has:

- two elements, say $0, 1$,
- an idempotent unary polynomial $\mathbf{e}_{01}(x)$ with range $\{0,1\}$,
- two binary polynomials $\wedge, \vee$ and a unary polynomial $\neg$ that act on the set $\{0,1\}$ like Boolean operation, i.e. meet, join and negation, respectively.

Let c be a constant bounding the sizes of all these four polynomials.

The presence of these polynomials allows us to translate each 3-CNF-SAT instance $\Phi \equiv \bigwedge_{i=1}^m \ell_1^i \vee \ell_2^i \vee \ell_3^i$, where $\ell_j^i \in \left\{z_i^j, \neg z_i^j\right\}$, into the equation of the algebra $\mathbf{A}$

$$\bigwedge_{j=1}^{m} \delta_1^j \mathbf{e}_{01}(x_1^j) \vee \delta_2^j \mathbf{e}_{01}(x_2^j) \vee \delta_3^j \mathbf{e}_{01}(x_3^j) = 1, \tag{1}$$

where

$$\delta_j^i \mathbf{e}_{01}(x_i^j) = \begin{cases} \mathbf{e}_{01}(x_i^j), & \text{if the literal } \ell_j^i \text{ is the variable, i.e., } \ell_j^i = z_i^j, \\ \neg \mathbf{e}_{01}(x_i^j), & \text{if } \ell_j^i \text{ is the negated variable, i.e., } \ell_j^i = \neg z_i^j. \end{cases}$$

It should be obvious that the formula Φ is satisfiable if and only if the equation (1) has a solution. However we need to take care of the size of this equation. But this can be secured by representing the m-ary conjunction in (1) in a balanced form, i.e. by a complete binary tree. This ensures us that the size of our equation is bounded by $O(c^{\log m})$, i.e. by a polynomial in m. This shows that in the presence of local Boolean behaviour in $\mathbf{A}$ the problem PolSat($\mathbf{A}$) is NP-complete, so that in view of ETH it cannot be in P.

Now we may assume that there is no local Boolean behaviour in $\mathbf{A}$, or in other words that the algebra $\mathbf{A}$ is solvable. In this case we simply refer to Theorem 1.2 to conclude that PolSat($\mathbf{A}$) $\in$ P implies $\mathsf{nr}\,(\mathbf{A}) \leqslant 2$, as claimed. ◀

By using the circuits constructed in Lemma 2.1 we can easily derive Theorem 1.1.

Proof of Theorem 1.1. The first part of the proof of Corollary 1.3 shows that the presence of Boolean local behaviour leads to NP-completeness of PolSat($\mathbf{A}$) and therefore also of CSat($\mathbf{A}$). Thus we may assume that $\mathbf{A}$ is solvable. In this case Lemma 2.2 supplies us with an element $a_{h-1} \in A$ so that each 3-CNF formula Φ can be turned, in a polynomial time, into a corresponding circuit $\mathbf{sat}_\Phi$ such that the equation $\mathbf{sat}_\Phi(\overline{x}) = a_{h-1}$ has a solution iff Φ is satisfiable.

This reduction obviously shows NP-completeness of CSat($\mathbf{A}$). ◀

3 Proofs of the Lemmas

For the proofs of Lemmas 2.1 and 2.2 we need an auxiliary Lemma. It uses the concept of the centralizer $(\alpha : \beta)$, that is the largest congruence θ satisfying $[\theta, \beta] \leqslant \alpha$.

▶ **Lemma 3.1.** *Let $\mathbf{A}$ be a finite solvable Malcev algebra and $\alpha \prec \beta$ be a covering pair of its congruences such that $\mathsf{nr}\,(\beta) > \mathsf{nr}\,(\alpha) > 0$. Then there is a join irreducible congruence γ with $\mathsf{nr}\,(\gamma) = \mathsf{nr}\,(\alpha)$ and $\alpha \leqslant (\gamma_- : \gamma)$ but $\beta \nleqslant (\gamma_- : \gamma)$.*

Moreover for any pair $(e', a') \in \gamma - \gamma_-$ and $(e, a) \notin (\gamma_- : \gamma)$ there is a binary polynomial $\mathbf{s}_{ea}(x, y)$ of $\mathbf{A}$, satisfying

$$\begin{aligned} &\mathbf{s}_{ea}(e', y) = e', \qquad \text{for all } y \in A, \\ &\mathbf{s}_{ea}(a', e) \overset{\gamma_-}{\equiv} e', \\ &\mathbf{s}_{ea}(a', a) = a'. \end{aligned} \tag{2}$$

Proof. Let $k = \mathsf{nr}\,(\beta) > \mathsf{nr}\,(\alpha) = k - 1$, i.e. $\beta \nleqslant \nu_{k-1} \geqslant \alpha$. Since $\mathsf{nr}\left(\beta^{(\omega)}\right) = \mathsf{nr}\,(\beta) - 1$ we know that $\beta^{(\omega)} \nleqslant \nu_{k-2}$ so there is a congruence φ such that $\beta^{(\omega)} \cap \nu_{k-2} \leqslant \varphi \prec \beta^{(\omega)}$. Put $\rho_0 = \beta^{(\omega)}$ and $\rho_{i+1} = [\rho_i, \alpha]$ and observe that $\rho_i \leqslant \alpha^{(i)}$ whenever $i \geqslant 1$. Since $\alpha^{(\omega)} = \alpha^{(j)}$ holds for some j, we have $\rho_j \leqslant \alpha^{(j)} = \alpha^{(\omega)} \leqslant \beta^{(\omega)} \cap \nu_{k-2} \leqslant \varphi$. As $\rho_0 = \beta^{(\omega)} \nleqslant \varphi$ then the minimal integer ℓ for which $\rho_\ell \leqslant \varphi$ is at least 1. Thus $\rho_{\ell-1} \vee \varphi = \beta^{(\omega)}$ so that in fact $\rho_1 = \left[\beta^{(\omega)}, \alpha\right] = [\rho_{\ell-1} \vee \varphi, \alpha] = [\rho_{\ell-1}, \alpha] \vee [\varphi, \alpha] = \rho_\ell \vee [\varphi, \alpha] \leqslant \varphi$. This shows $\alpha \leqslant \left(\varphi : \beta^{(\omega)}\right)$. Obviously $\beta \nleqslant \left(\varphi : \beta^{(\omega)}\right)$ as $\left[\beta^{(\omega)}, \beta\right] = \beta^{(\omega)} \nleqslant \varphi$. Now we pick a minimal congruence γ below $\beta^{(\omega)}$ but not below φ. Obviously γ is join irreducible and the covering pair $\varphi \prec \beta^{(\omega)}$ transposes down to $\gamma_- \prec \gamma$ where γ_- is the unique subcover of γ. Consequently $(\gamma_- : \gamma) = \left(\varphi : \beta^{(\omega)}\right)$, i. e. γ has the properties described in the Lemma. To calculate $\mathsf{nr}\,(\gamma)$ note first that $\gamma \leqslant \beta^{(\omega)} \leqslant \nu_{k-1}$ so that $\mathsf{nr}\,(\gamma) \leqslant k - 1$. But $\mathsf{nr}\,(\gamma) \leqslant k - 2$ would give $\gamma \leqslant \nu_{k-2} \cap \beta^{(\omega)} \leqslant \varphi$, contrary to our choice of γ.

For the second part of the Lemma we fix any pair $(e',a') \in \gamma - \gamma_-$. Due to the join irreducibility of γ we know that $\gamma = \Theta(e',a')$. Now if $(e,a) \notin (\gamma_- : \gamma)$ then $[\Theta(e,a), \Theta(e',a')] \nleqslant \gamma_-$ so that Exercise 6.6 in [6] supplies us with a binary polynomial $\mathbf{s}(x,y)$ of $\mathbf{A}$ such that

$$\mathbf{s}(e',e) \overset{\gamma_-}{\equiv} \mathbf{s}(a',e),$$

$$\mathbf{s}(e',a) \overset{\gamma_-}{\not\equiv} \mathbf{s}(a',a).$$

The very last line gives $\Theta(\mathbf{s}(e',a), \mathbf{s}(a',a)) = \gamma \ni (e',a')$ and therefore there is a unary polynomial $\mathbf{p}$ of $\mathbf{A}$ that takes the pair $(\mathbf{s}(e',a), \mathbf{s}(a',a))$ to (e',a'). Using Malcev polynomial $\mathbf{d}$ we modify $\mathbf{s}(x,y)$ to a new polynomial $\mathbf{s}_{ea}(x,y) = \mathbf{d}(\mathbf{ps}(x,y), \mathbf{ps}(e',y), e')$ for which it should be easy to check that

$$\begin{array}{lclclcl} \mathbf{s}_{ea}(e',y) & = & \mathbf{d}(\mathbf{ps}(e',y), \mathbf{ps}(e',y), e') & = & e', & & \\ \mathbf{s}_{ea}(a',e) & = & \mathbf{d}(\mathbf{ps}(a',e), \mathbf{ps}(e',e), e') & \overset{\gamma_-}{\equiv} & \mathbf{d}(\mathbf{ps}(e',e), \mathbf{ps}(e',e), e') & = & e', \\ \mathbf{s}_{ea}(a',a) & = & \mathbf{d}(\mathbf{ps}(a',a), \mathbf{ps}(e',a), e') & = & \mathbf{d}(a', e', e') & = & a', \end{array}$$

so that all three conditions of (2) hold. ◀

With the help of Lemma 3.1 we are ready to prove Lemmas 2.1 and 2.2.

Proof of Lemma 2.1. We start our proof by referring to Lemma 3.1 with $(\alpha,\beta) = (\delta_-,\delta)$ to get a join irreducible congruence γ with $\delta \nleqslant (\gamma_- : \gamma) \geqslant \delta_-$. We fix a (γ_-,γ)-minimal set V of $\mathbf{A}$ and a pair $(e',a') \in \gamma|_V - \gamma_-$. On the other hand $\delta = \Theta(e,a)$ ensures us that $(e,a) \notin (\gamma_- : \gamma)$ so that Lemma 3.1 supplies us with a polynomial $\mathbf{s}_{ea}(x,y)$ satisfying (2). Our first goal is to consecutively modify this polynomial $\mathbf{s}_{ea}$ to force the middle line in the display (2) to be the real equality instead of $\overset{\gamma_-}{\equiv}$. First we replace $\mathbf{s}_{ea}(x,y)$ with $\mathbf{e}_V\mathbf{s}_{ea}(\mathbf{e}_V(x),y)$, where $\mathbf{e}_V$ is a unary idempotent polynomial of $\mathbf{A}$ with range V. This new $\mathbf{s}_{ea}(x,y)$ not only satisfies (2) but also has its range contained in V and for any fixed y we have $\mathbf{s}_{ea}(A,y) = \mathbf{s}_{ea}(V,y)$.

Now, note that the first two lines of (2) tell us that the unary polynomial $\mathbf{s}^0(x) = \mathbf{s}_{ea}(x,e)$ does not permute V and consequently $\mathbf{s}^0(A) = \mathbf{s}^0(V) \subsetneq V$. Note also that for all $\varphi \prec \psi \leqslant \gamma$ we have $\mathbf{s}^0(\psi) \subseteq \varphi$, as otherwise the range of $\mathbf{s}^0$ would contain an (φ,ψ)-minimal set properly contained in V. This however (in view of Lemma 4.30 of [9]) cannot happen as V is a minimal set of type **2**. Now if a maximal chain of congruences strictly below γ has exactly l congruences then by replacing the polynomial $\mathbf{s}_{ea}(x,y)$ with $\mathbf{s}_{ea}(\ldots\mathbf{s}_{ea}(\mathbf{s}_{ea}(x,y),y)\ldots,y)$, where the iteration in the variable x is done l times, we keep the first and the third line of (2) to be true, while the middle one can be replaced by the equality. Thus we end up with a new polynomial $\mathbf{s}_{ea}(x,y)$ satisfying

$$\begin{array}{ll} \mathbf{s}_{ea}(e',y) = e', & \text{for all } y \in A, \\ \mathbf{s}_{ea}(a',e) = e', & \\ \mathbf{s}_{ea}(a',a) = a'. & \end{array} \tag{3}$$

Now the n-ary polynomial $\mathbf{and}^0_n(x_1,\ldots,x_n) = \mathbf{s}_{ea}(\ldots\mathbf{s}_{ea}(\mathbf{s}_{ea}(a',x_1),x_2)\ldots,x_n)$ satisfies

$$\mathbf{and}^0_n(x_1,\ldots,x_n) = \begin{cases} a', & \text{if } x_1 = \ldots = x_n = a, \\ e', & \text{otherwise,} \end{cases} \tag{4}$$

whenever $x_1,\ldots,x_n \in \{e,a\}$.

To conclude our argument note that $\gamma \nleqslant \nu_{\mathsf{nr}(\delta_-)-1}$ and then simply pick a minimal congruence $\delta^\star$ below γ but not below $\nu_{\mathsf{nr}(\delta_-)-1}$. Obviously $\delta^\star$ is join irreducible (with the unique subcover $\delta^\star_-$) so that it is principal, say $\delta^\star = \Theta(e^\star,a^\star)$. Also, by minimality we have $\mathsf{nr}\left(\delta^\star\right) = \mathsf{nr}\left(\delta_-\right)$ and $\mathsf{nr}\left(\delta^\star_-\right) = \mathsf{nr}\left(\delta_-\right) - 1$.

Finally $(e^\star, a^\star) \in \delta^\star \subseteq \gamma = \Theta(e', a')$, so that there is a unary polynomial $\mathbf{p}$ of $\mathbf{A}$ that maps e' onto $e^\star$ and a' onto $a^\star$. By (4) it should be clear that the polynomial $\mathbf{and}_n(x_1, \ldots, x_n) = \mathbf{p}(\mathbf{and}^0_n(x_1, \ldots, x_n))$ does the job described in the Lemma.

To bound the length of the polynomial $\mathbf{and}_n$ and the size of the circuits that compute $\mathbf{and}_n$, first note that the polynomial $\mathbf{s}_{ea}$ can be realized by a circuit of a constant size (independent of n). Thus the entire circuit computing $\mathbf{and}^0_n$, and therefore of $\mathbf{and}_n$, can be constructed in a linear time $O(n)$. On the other hand unwinding this circuit to the tree (corresponding to the polynomial $\mathbf{and}_n$) enlarges the size exponentially and requires at most exponential time $2^{O(n)}$. ◀

The proof of Lemma 2.2 is slightly more involved.

Proof of Lemma 2.2. We start with observing that $\nu_{h-1} < 1_\mathbf{A}$ so that we can pick a cover $\beta \succ \nu_{h-1}$. This together with α set to ν_{h-1} allows us to use Lemma 3.1 to produce a join irreducible congruence γ with $\nu_{h-1} \leqslant (\gamma_- : \gamma) \not\geqslant \beta$. In particular we know that $(\gamma_- : \gamma) \neq 1_\mathbf{A}$.

By Lemma 3.1 we know that $\mathsf{nr}\,(\gamma) = \mathsf{nr}\,(\nu_{h-1}) = h - 1$ so that $\gamma \nleqslant \nu_{h-2}$ and we can pick δ^{h-1} to be a minimal congruence below γ but not below ν_{h-2}. Obviously δ^{h-1} is join irreducible so that it is a principal congruence, say $\delta^{h-1} = \Theta(e_{h-1}, a_{h-1})$. By the very same token we know that γ is principal, but here we need to choose its generating pair more carefully. First we fix a (γ_-, γ)-minimal set V and then a pair $(e', a') \in \gamma|_V - \gamma_-$ to have $\gamma = \Theta(e', a')$. The (γ_-, γ)-trace of V containing both e' and a' is denoted by N. We know that the induced algebra $(\mathbf{A}|_N)/\gamma_-$ is polynomially equivalent to a (one dimensional) vector space and we may assume that e'/γ_- is its zero element with respect to the vectors addition $+$ which has to be a polynomial of $\mathbf{A}$. Since $\Theta(e_{h-1}, a_{h-1}) = \delta^{h-1} \leqslant \gamma = \Theta(e', a')$ we can pick a unary polynomial $\mathbf{p}$ of $\mathbf{A}$ that maps e' to e_{h-1} and a' to a_{h-1}. This polynomial is going to be used at the end of the proof.

Now we put $\tau = (\gamma_- : \gamma)$ and choose a transversal $\{d_0, d_1, \ldots, d_r\}$ of A/τ. If $i \neq j$ then $(d_i, d_j) \notin (\gamma_- : \gamma)$ and Lemma 3.1 gives us a binary polynomial $\mathbf{s}_{ij} = \mathbf{s}_{d_i,d_j}$ satisfying

$$\begin{aligned} \mathbf{s}_{ij}(e', y) &= e', \qquad \text{for all } y \in A, \\ \mathbf{s}_{ij}(a', d_i) &\overset{\gamma_-}{\equiv} e', \\ \mathbf{s}_{ij}(a', d_j) &= a'. \end{aligned} \tag{5}$$

As in the proof of Lemma 2.1 we replace $\mathbf{s}_{ij}(x, y)$ by $\mathbf{e}_V \mathbf{s}_{ij}(\mathbf{e}_V(x), y)$, where $\mathbf{e}_V$ is the unary idempotent polynomial of $\mathbf{A}$ with the range V. Obviously the properties in the display (5) hold for this new $\mathbf{s}_{ij}$, but this new polynomial has the range contained in V and for any fixed $y \in A$ the mapping $V \ni v \longmapsto \mathbf{s}_{ij}(v, y) \in V$ is either a permutation of V or collapses $\gamma|_V$ to γ_-, i.e. it is constant modulo γ_- on $\gamma|_V$-classes. Thus, iterating $\mathbf{s}_{ij}(v, y)$ in the first variable a sufficient number of times, we can modify $\mathbf{s}_{ij}$ to additionally have that (for each fixed $y \in A$) the new polynomial $\mathbf{s}_{ij}(v, y)$ is either the identity map on V or it is constant modulo γ_- on $\gamma|_V$-classes. Actually, in the second case, i.e. if $\mathbf{s}_{ij}(v, y)$ collapses $\gamma|_V$ to γ_-, it collapses the entire trace N to $\mathbf{s}_{ij}(e', y)/\gamma_- = e'/\gamma_-$. Summing up, we produced polynomials $\mathbf{s}_{ij}$ satisfying

$$\begin{array}{rcll} \mathbf{s}_{ij}(e', y) & = & e', & \text{for each } \ y \in A, \\ \mathbf{s}_{ij}(v, d_i) & \overset{\gamma_-}{\equiv} & e', & \text{for each } \ v \in N, \\ \mathbf{s}_{ij}(v, d_j) & = & v, & \text{for each } \ v \in V. \end{array}$$

Now, using the fact that $[\gamma,\tau] \leqslant \gamma_-$ we can keep the above equalities modulo γ_- by varying the second variable modulo τ:

$$\begin{array}{rcll} \mathbf{s}_{ij}(e',y) & = & e', & \text{for each } \ y \in A, \\ \mathbf{s}_{ij}(v,y) & \overset{\gamma_-}{\equiv} & e', & \text{for each } \ v \in N \text{ and } \ y \in d_i/\tau, \\ \mathbf{s}_{ij}(v,y) & \overset{\gamma_-}{\equiv} & v, & \text{for each } \ v \in V \text{ and } \ y \in d_j/\tau. \end{array} \tag{6}$$

Now for each $j = 0,\dots,r$ define

$$\mathbf{s}_j(x,y) = \mathbf{s}_{i_1 j}(\dots \mathbf{s}_{i_{r-1} j}(\mathbf{s}_{i_r j}(x,y),y)\dots,y),$$

where $\{j, i_1,\dots,i_r\} = \{0,1,\dots,r\}$. Obviously $\mathbf{s}_j$ has the range contained in V and

$$\begin{array}{rcll} \mathbf{s}_{j}(e',y) & = & e', & \text{for each } \ y \in A, \\ \mathbf{s}_{j}(v,y) & \overset{\gamma_-}{\equiv} & e', & \text{for each } \ v \in N \text{ and } \ y \in A - d_j/\tau, \\ \mathbf{s}_{j}(v,y) & \overset{\gamma_-}{\equiv} & v, & \text{for each } \ v \in V \text{ and } \ y \in d_j/\tau. \end{array} \tag{7}$$

Indeed, the first and the last item follow directly from the definition of $\mathbf{s}_j$. To see the middle one, note that for $v \in N$ and $y \in d_{i_\ell}/\tau$, defining $v' = \mathbf{s}_{i_{\ell+1}}(\dots \mathbf{s}_{i_{r-1} j}(\mathbf{s}_{i_r j}(v,y),y)\dots,y)$ we have

$$v' \overset{\gamma}{\equiv} \mathbf{s}_{i_{\ell+1} j}(\dots \mathbf{s}_{i_{r-1} j}(\mathbf{s}_{i_r j}(e',y),y)\dots,y) = e',$$

i.e. $v' \in N$ so that $\mathbf{s}_{i_\ell j}(v',y) \overset{\gamma_-}{\equiv} e'$, and consequently

$$\mathbf{s}_j(v,y) = \mathbf{s}_{i_1 j}(\dots \mathbf{s}_{i_{\ell-1} j}(\mathbf{s}_{i_\ell j}(v',y),y)\dots,y) \overset{\gamma_-}{\equiv} \mathbf{s}_{i_1 j}(\dots \mathbf{s}_{i_{\ell-1} j}(e',y)\dots,y) = e'.$$

Recall that $(\mathbf{A}|_N)/\gamma_-$ is polynomially equivalent to a vector space in which e'/γ_- serves as a zero element, while the addition is defined by $x+y = \mathbf{d}(x,e',y)$ and $x-y = \mathbf{d}(x,y,e')$. Obviously this addition does not behave so nice outside the trace N and before factoring out by γ_- but since for $v \in N$ (and arbitrary $y \in A$) the elements $\mathbf{s}_j(v,y)$ are in $\mathbf{A}|_N$, it makes sense to sum them up and define

$$\mathbf{s}_\top(x,y) = \sum_{j=1}^{r} \mathbf{s}_j(x,y)$$

(by associating the "summands" to the left) to observe that

$$\begin{array}{rcll} \mathbf{s}_{\top}(e',y) & = & e', & \text{for each } \ y \in A, \\ \mathbf{s}_{\top}(v,y) & \overset{\gamma_-}{\equiv} & e', & \text{for each } \ v \in N \text{ and } \ y \in d_0/\tau, \\ \mathbf{s}_{\top}(v,y) & \overset{\gamma_-}{\equiv} & v, & \text{for each } \ v \in N \text{ and } \ y \in A - d_0/\tau. \end{array} \tag{8}$$

To see (8) first note that in the sum defining $\mathbf{s}_\top$ for $v \in N$, at most one summand lies outside of e'/γ_-, namely $\mathbf{s}_j(v,y)$ with $j \neq 0$ for which $y \in d_j/\tau$. This obviously gives the last two lines in (8) as well as $\mathbf{s}_\top(e',y) \overset{\gamma_-}{\equiv} e'$. To replace this by the equality note that due to the first line in (7) all summands in $\mathbf{s}_\top(e',y)$ are equal to e' so that "summing" them up with the help of Malcev polynomial $\mathbf{d}$ returns e'.

Now we put $\mathbf{s}_\perp(x,y) = \mathbf{s}_0(x,y)$ and observe that for any fixed $v \in N$ both $\mathbf{s}_\top(v,y)$ and $\mathbf{s}_\perp(v,y)$ have their ranges contained in $e'/\gamma_- \cup v/\gamma_-$. Moreover $\mathbf{s}_\top$ and $\mathbf{s}_\perp$ are complementary in the sense that $\mathbf{s}_\top(v,y) + \mathbf{s}_\perp(v,y) = v$. In fact they switch their values (between v and e') depending on whether y is in $A_\perp = d_0/\tau$ or in its complement $A_\top = A - A_\perp = \bigcup_{i=1}^{r} d_i/\tau$.

Now, for $\overline{\varepsilon} = (\varepsilon_1, \varepsilon_2, \varepsilon_3) \in \{\top, \bot\}^3$ we define the polynomial

$$\begin{aligned}\breve{\mathbf{s}}_{\overline{\varepsilon}}(v, y_1, y_2, y_3) = {} & \mathbf{s}_{\varepsilon_3}(\mathbf{s}_{\varepsilon_2}(\mathbf{s}_{\varepsilon_1}(v, y_1), y_2), y_3) + \mathbf{s}_{\varepsilon_1}(v, y_1) + \mathbf{s}_{\varepsilon_2}(v, y_2) + \mathbf{s}_{\varepsilon_3}(v, y_3) \\ & - \mathbf{s}_{\varepsilon_2}(\mathbf{s}_{\varepsilon_1}(v, y_1), y_2) - \mathbf{s}_{\varepsilon_3}(\mathbf{s}_{\varepsilon_1}(v, y_1), y_3) - \mathbf{s}_{\varepsilon_2}(\mathbf{s}_{\varepsilon_3}(v, y_2), y_3),\end{aligned}$$

Using (7) and (8) one can calculate that

$$\begin{array}{rcll}\breve{\mathbf{s}}_{\overline{\varepsilon}}(e', y_1, y_2, y_3) & = & e', & \\ \breve{\mathbf{s}}_{\overline{\varepsilon}}(v, y_1, y_2, y_3) & \overset{\gamma_-}{\equiv} & e', & \text{if } y_i \notin A_{\varepsilon_i} \text{ for } i \in \{1,2,3\}, \\ \breve{\mathbf{s}}_{\overline{\varepsilon}}(v, y_1, y_2, y_3) & \overset{\gamma_-}{\equiv} & v, & \text{if } y_i \in A_{\varepsilon_i} \text{ for some } i,\end{array} \tag{9}$$

due to the fact that, modulo γ_-, each of the seven summands in $\breve{\mathbf{s}}_{\overline{\varepsilon}}(v, y_1, y_2, y_3)$ is either e', or v, or $-v$.
Arguing like in the proof of Lemma 2.1, by going down from γ_- to 0 through a maximal chain of congruences (and iterating $\breve{\mathbf{s}}_{\overline{\varepsilon}}$ in the first variable), we improve (9) to get

$$\begin{array}{rcll}\breve{\mathbf{s}}_{\overline{\varepsilon}}(e', y_1, y_2, y_3) & = & e', & \\ \breve{\mathbf{s}}_{\overline{\varepsilon}}(v, y_1, y_2, y_3) & = & e', & \text{if } y_i \notin A_{\varepsilon_i} \text{ for every } i \in \{1,2,3\}, \\ \breve{\mathbf{s}}_{\overline{\varepsilon}}(v, y_1, y_2, y_3) & = & v, & \text{if } y_i \in A_{\varepsilon_i} \text{ for some } i.\end{array} \tag{10}$$

With the help of the polynomials $\breve{\mathbf{s}}_{\overline{\varepsilon}}$ (which have been invented to code clauses) we can create the polynomial $\mathbf{sat}_\Phi$ for a given 3-CNF formula $\Phi \equiv \bigwedge_{j=1}^m \ell_1^j \vee \ell_2^j \vee \ell_3^j$, where $\ell_i^j \in \left\{z_i^j, \neg z_i^j\right\}$, by first putting

$$\varepsilon_i^j = \begin{cases} \top, & \text{if the literal } \ell_i^j \text{ is the variable, i.e., } \ell_i^j = z_i^j, \\ \bot, & \text{if the literal } \ell_i^j \text{ is the negated variable, i.e., } \ell_i^j = \neg z_i^j, \end{cases}$$

then $\overline{\varepsilon}^j = (\varepsilon_1^j, \varepsilon_2^j, \varepsilon_3^j)$ and finally

$$\mathbf{sat}_\Phi(x_1^1, x_2^1, x_3^1, \dots, x_1^m, x_2^m, x_3^m) = \mathbf{p}(\breve{\mathbf{s}}_{\overline{\varepsilon}^m}(\dots \breve{\mathbf{s}}_{\overline{\varepsilon}^1}(a', x_1^1, x_2^1, x_3^1), \dots x_1^m, x_2^m, x_3^m)).$$

By (10) it should be clear that for any evaluation of the z_i^j's in $\{\top, \bot\}$ and x_i^j's in A so that $x_i^j \in A_{z_i^j}$ we have

$$\Phi(z_1^1, z_2^1, z_3^1, \dots, z_1^m, z_2^m, z_3^m) = \top \quad \text{iff} \quad \mathbf{sat}_\Phi(x_1^1, x_2^1, x_3^1, \dots, x_1^m, x_2^m, x_3^m) = a_{h-1}.$$

as required by the Lemma.

The complexity arguments simply repeat those from the proof of Lemma 2.1 to bound the time needed to construct the polynomial $\mathbf{sat}_\Phi$ by $2^{O(m)}$, while to construct the circuit computing $\mathbf{sat}_\Phi$ by $O(m)$. ◀

4 Conclusions and Open Problems

Since Theorem 1.1 that improves the result from [18], one can hope for a classification of finite algebras from congruence modular varieties that have tractable CSat. Indeed, by [18, Corollary 6.5], such an algebra has to decompose into a direct product $\mathbf{S} \times \mathbf{L}$ of a solvable algebra $\mathbf{S}$ and an algebra $\mathbf{L}$ that behaves pretty similarly to a lattice (at least locally). Now, our Theorem 1.1 forces $\mathbf{S}$ to be nilpotent without actually assuming (like it has been done in [18]) that CSat is tractable for all quotients of $\mathbf{S}$. The paper [18] also enforces that the algebra $\mathbf{L}$ has to behave not only like a lattice, but in fact like a distributive lattice, provided CSat is tractable for all quotients of $\mathbf{L}$. The natural problem is to eliminate this strong assumption about quotients also from the $\mathbf{L}$ side, i.e. for algebras from congruence distributive varieties. Thus, we are left with the following

▶ **Question 1.** *Does tractability of CSat for a finite algebra $\mathbf{L}$ from a congruence distributive variety implies tractability of CSat for all quotients of $\mathbf{L}$?*

As we have already mentioned in the Introduction the classification of finite algebras with tractable CSat is not fully done on the solvable side. Now, by Theorem 1.1, we know that such an algebra $\mathbf{S}$ has to be nilpotent. Moreover, due to the work of Kompatscher [23], we know that in fact $\mathsf{sr}(\mathbf{A}) \leqslant 2$. Unfortunately this bound does not suffice to have tractable CSat($\mathbf{S}$), as it has been shown by some examples described in [17]. On the other hand one cannot hope to strengthen this bound and force $\mathbf{S}$ to be supernilpotent (in which case [18] gives a polynomial time algorithm for CSat). This in turn has been witnessed by our examples provided in [15]. Thus we are left with the following

▶ **Problem 2.** *Characterize finite nilpotent Malcev algebras of supernilpotent rank at most* 2 *with tractable CSat.*

A very similar, and somehow connected, problem CEqv($\mathbf{A}$) of circuit equivalence has also been considered in [18]. This time we ask if two circuits over $\mathbf{A}$ compute the same function. In this case, if a finite algebra $\mathbf{A}$ is taken from a congruence modular variety then [18] shows that tractability of CEqv($\mathbf{A}$) implies that $\mathbf{A}$ is solvable. Note that there is no lattice-like part here, as the co-NP-completeness of CEqv even over the 2-element lattice eliminates this part. Arguing like in the proof of Theorem 1.1 we can force nilpotency of algebras with tractable CEqv. Again, Kompatscher [23] forces such algebras to have supernilpotent rank bounded by 2. Suprisingly, here we do not have any single example of a nilpotent algebra $\mathbf{A}$ with $\mathsf{sr}(\mathbf{A}) = 2$ and intractable CEqv($\mathbf{A}$). In fact among the examples in [17] of algebras with $\mathsf{sr}(\mathbf{A}) = 2$ but intractable CSat($\mathbf{A}$) there are 2-nilpotent algebras. However, [19] shows that for 2-nilpotent algebras the problem CEqv is tractable. Therefore the answer to the next Problem differs from the answer to Problem 2.

▶ **Problem 3.** *Characterize finite nilpotent Malcev algebras of supernilpotent rank at most* 2 *with tractable CEqv.*

In our opinion this difference in the complexity for CSat and CEqv may result in a search for completely new techniques. Nevertheless, note that both CSat and CEqv behave the same on supernilpotent algebras, as CEqv for such algebras has been shown to be tractable in [1].

Now we switch to the problem PolSat. For a fixed algebra its complexity is not bigger than this of CSat. However, as there are examples (e.g. the already mentioned groups $\mathbf{S}_3$ or $\mathbf{A}_4$) where CSat is essentially harder. This is because using (in CSat) additional polynomials we compress (in our opinion artificially inflated) the input of PolSat. For a better understanding of this phenomenon among Malcev algebras first note that, due to Theorem 1.2, PolSat($\mathbf{A}$) $\in \mathsf{P} \not\ni$ CSat($\mathbf{A}$) may happen only if $\mathsf{nr}(\mathbf{A}) \leqslant 2$. Among algebras of nilpotent rank 1, i.e. among nilpotent algebras, all known cases (i.e. both tractable and intractable examples, as well as some more general results, like for supernilpotent algebras) enjoy the same complexity for both PolSat and CSat. This leads to the following:

▶ **Question 4.** *Does there exist a finite nilpotent Malcev algebra with tractable PolSat and intractable CSat?*

A careful reading of the proof of Corollary 6.5 in [18] shows that the earlier described decomposition $\mathbf{A} = \mathbf{S} \times \mathbf{L}$ requires in fact a weaker assumption that only PolSat(A) (but not necessarily CSat) is not NP-complete. Note that a positive answer to Question 1 can possibly be carried out to PolSat. Thus we believe that the next question, similar to Question 4, has a negative answer.

▶ **Question 5.** *Does there exist a finite algebra from a congruence distributive variety with tractable* PolSat *and intractable* CSat*?*

Summing up we expect that the following conjecture holds.

▶ **Conjecture 6.** *The only examples of finite algebras (from a congruence modular variety) that separate complexity of* PolSat *and* CSat *have to be solvable and of nilpotent rank* 2*.*

When considering complexity of i.e. the polynomial equivalence problem, versus this of CEqv we can repeat Questions 4 and Conjecture 6. An analogous modification of Question 5 is already answered, as both PolEqv and CEqv are co-NP-complete for nontrivial algebras in this realm.

References

1 Erhard Aichinger and Nebojša Mudrinski. Some applications of higher commutators in Mal'cev algebras. *Algebra Universalis*, 63(4):367–403, 2010. doi:10.1007/s00012-010-0084-1.

2 David Mix Barrington, Pierre McKenzie, Cris Moore, Pascal Tesson, and Denis Thérien. Equation satisfiability and program satisfiability for finite monoids. In *25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000)*, pages 172–181. Springer Berlin Heidelberg, 2000. doi:10.1007/3-540-44612-5_13.

3 Stanley Burris and Hanamantagouda Sankappanavar. *A Course in Universal Algebra With 36 Illustrations*. Springer-Verlag, New York, 1981.

4 Stephen Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158, 1971. doi:10.1145/800157.805047.

5 Attila Földvári and Gábor Horváth. The complexity of the equation solvability and equivalence problems over finite groups. *International Journal of Algebra and Computation*, 30(03):1–17, 2019. doi:10.1142/S0218196720500137.

6 Ralph Freese and Ralph McKenzie. *Commutator Theory For Congruence Modular Varieties*. London Mathematical Society Lecture Notes, No. 125. Cambridge University Press, 1987.

7 Mikael Goldmann and Alexander Russell. The complexity of solving equations over finite groups. *Information and Computation*, 178(1):253–262, 2002. doi:10.1006/inco.2002.3173.

8 Tomasz Gorazd and Jacek Krzaczkowski. Term equation satisfiability over finite algebras. *International Journal of Algebra and Computation*, 20(08):1001–1020, 2010. doi:10.1142/S021819671000600X.

9 David Hobby and Ralph McKenzie. *Structure of Finite Algebras*. Contemporary Mathematics vol. 76. American Mathematical Society, 1988. doi:10.1090/conm/076.

10 Gábor Horváth. The complexity of the equivalence and equation solvability problems over nilpotent rings and groups. *Algebra Universalis*, 66(4):391–403, 2011. doi:10.1007/s00012-011-0163-y.

11 Gábor Horváth. The complexity of the equivalence and equation solvability problems over meta-abelian groups. *Journal of Algebra*, 433:208–230, 2015. doi:10.1016/j.jalgebra.2015.03.015.

12 Gábor Horváth and Csaba Szabó. The complexity of checking identities over finite groups. *International Journal of Algebra and Computation*, 16(05):931–940, 2006. doi:10.1142/S0218196706003256.

13 Gábor Horváth and Csaba Szabó. Equivalence and equation solvability problems for the alternating group A_4. *Journal of Pure and Applied Algebra*, 216(10):2170–2176, 2012. doi:10.1016/j.jpaa.2012.02.007.

14 Paweł Idziak, Piotr Kawałek, Jacek Krzaczkowski, and Armin Weiß. Equation satisfiability in solvable groups. To appear in *Theory of Computing Systems*, 2021. arXiv:2010.11788.

15 Paweł Idziak, Piotr Kawałek, and Jacek Krzaczkowski. Expressive power, satisfiability and equivalence of circuits over nilpotent algebras. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117, pages 17:1–17:15. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.17.

16 Paweł Idziak, Piotr Kawałek, and Jacek Krzaczkowski. Intermediate problems in modular circuits satisfiability. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2020)*, pages 578–590, 2020. doi:10.1145/3373718.3394780.

17 Paweł Idziak, Piotr Kawałek, and Jacek Krzaczkowski. Complexity of modular circuits, 2021. arXiv:2106.02947.

18 Paweł Idziak and Jacek Krzaczkowski. Satisfiability in multi-valued circuits. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2018)*, pages 550–558, 2018. Full version: to appear in SIAM Journal on Computing (see also 1710.08163). doi:10.1145/3209108.3209173.

19 Piotr Kawałek, Michael Kompatscher, and Jacek Krzaczkowski. Circuit equivalence in 2-nilpotent algebras, 2019. arXiv:1909.12256.

20 Keith Kearnes. Congruence modular varieties with small free spectra. *Algebra Universalis*, 42(3):165–181, 1999. doi:10.1007/s000120050132.

21 Ondrej Klíma. Complexity issues of checking identities in finite monoids. *Semigroup Forum*, 79(3):435–444, 2009. doi:10.1007/s00233-009-9180-y.

22 Michael Kompatscher. The equation solvability problem over supernilpotent algebras with mal'cev term. *International Journal of Algebra and Computation*, 28(06):1005–1015, 2018. doi:10.1142/S0218196718500443.

23 Michael Kompatscher. CSAT and CEQV for nilpotent Maltsev algebras of fitting length > 2, 2021. arXiv:2105.00689.

24 Yuri Matiyasevich. Enumerable sets are diophantine. In *Soviet Mathematics Doklady*, volume 11, pages 354–358, 1970.

25 Bernhard Schwarz. The complexity of satisfiability problems over finite lattices. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS 2004)*, pages 31–43. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-24749-4_4.

26 Armin Weiß. Hardness of equations over finite solvable groups under the exponential time hypothesis. In *Proceedings of 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, 2020. doi:10.4230/LIPIcs.ICALP.2020.102.

Classes of Intersection Digraphs with Good Algorithmic Properties

Lars Jaffke ✉ 🏠

Department of Informatics, University of Bergen, Norway

O-joung Kwon ✉ 🏠

Department of Mathematics, Incheon National University, South Korea
Institute for Basic Science, South Korea

Jan Arne Telle ✉ 🏠

Department of Informatics, University of Bergen, Norway

Abstract

While intersection graphs play a central role in the algorithmic analysis of hard problems on undirected graphs, the role of intersection *digraphs* in algorithms is much less understood. We present several contributions towards a better understanding of the algorithmic treatment of intersection digraphs. First, we introduce natural classes of intersection digraphs that generalize several classes studied in the literature. Second, we define the directed locally checkable vertex (DLCV) problems, which capture many well-studied problems on digraphs such as (Independent) Dominating Set, Kernel, and H-Homomorphism. Third, we give a new width measure of digraphs, *bi-mim-width*, and show that the DLCV problems are polynomial-time solvable when we are provided a decomposition of small bi-mim-width. Fourth, we show that several classes of intersection digraphs have bounded bi-mim-width, implying that we can solve all DLCV problems on these classes in polynomial time given an intersection representation of the input digraph. We identify reflexivity as a useful condition to obtain intersection digraph classes of bounded bi-mim-width, and therefore to obtain positive algorithmic results.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases intersection digraphs, H-digraphs, reflexive digraphs, directed domination, directed H-homomorphism

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.38

Related Version *Full Version*: https://arxiv.org/abs/2105.01413

Funding *Lars Jaffke*: Supported by the Norwegian Research Council via project 274526.
O-joung Kwon: Supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. NRF-2018R1D1A1B07050294) and by Institute for Basic Science (IBS-R029-C1).

1 Introduction

The computational intractability of graph problems is often dealt with by restricting the input graph to be a member of some graph class and exploit the structural properties of this class to design efficient algorithms. Intersection graph classes are an extensively studied family of classes of undirected graphs where vertices are represented by sets with two vertices being adjacent if and only if their corresponding sets intersect. For instance, a graph is an *interval graph* if it is an intersection graph of intervals on a line. The literature on algorithmic aspects of classes of intersection graphs is vast, and we refer to [13] for an overview. Even though the concept of intersection *digraphs* has already been introduced in the early 1980s [9], these classes of directed graphs have not received nearly as much

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 38; pp. 38:1–38:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

attention in the algorithmic literature as their undirected counterparts. That is not to say that they have not been considered before; for instance, interval digraphs [38], circular-arc digraphs [39], and permutation digraphs [33] have been introduced quite early on.

Formally, a digraph G is an *intersection digraph* if there exists a family $\{(S_v, T_v) : v \in V(G)\}$ of ordered pairs of sets such that there is an edge from v to w in G if and only if S_v intersects T_w. Note that we add a loop on a vertex v if S_v and T_v intersect. Even for interval digraphs, a natural starting point for the investigation of algorithmic properties of intersection digraphs, no algorithmic applications are known besides a polynomial-time recognition algorithm of the class [33]. One possible explanation for this is that the class of interval digraphs appears to be much richer than their undirected counterparts. We observe that interval digraphs contain, for each integer n, some orientation of the $(n \times n)$-grid (see Proposition 15); in contrast, interval graphs do not contain an induced subgraph isomorphic to the 1-subdivision of the claw. This shows that the underlying undirected graphs of interval digraphs are very different from interval graphs.

The case of interval digraphs suggests that further structural restrictions are necessary to make classes of intersection digraphs amenable for algorithmic treatment. In this vein, restrictions of interval digraphs have been considered in the literature [20, 35] with applications to digraph problems such as Independent Dominating Set, Kernel, and List Homomorphism. A common feature of the restrictions considered in [20, 35] is that the digraphs are *reflexive*, meaning that each vertex has a loop. Note that for a class of intersection digraphs, reflexivity gives much more additional structure than just added loops.

In this work, we give a host of algorithmic applications of intersection digraph classes, in the following manner:

- We give new and more general classes of intersection digraphs, namely H-digraphs, rooted directed path digraphs, and H-convex digraphs. (See the discussion below Theorem 3 for definitions.)
- We introduce directed analogues of the locally checkable vertex problems [43], which include many well-studied digraph problems such as (Independent) Dominating Set, Kernel, H-Homomorphism, and Oriented k-Coloring, see Tables 1 and 2.
- We define a new width measure of digraphs, called *bi-mim-width*, and prove that the directed locally checkable vertex problems can be solved in polynomial time when a decomposition of bounded bi-mim-width of the input graph is given.
- We prove that fairly general subclasses of these intersection digraph classes have bounded bi-mim-width, see Figure 1.

Note in particular that the last item implies that given a representation of the input digraph, all directed locally checkable problems are solvable in polynomial time on the classes of intersection digraphs in question. For H-digraphs, we identify reflexivity as the additional restriction that gives bounded bi-mim-width, and therefore algorithmic applications, while we prove that the bi-mim-width is unbounded when we drop this requirement. Recently, Francis, Hell, and Jacob [22] obtained polynomial-time algorithms for Kernel, Dominating Set, and Absorbing Set on reflexive interval digraphs. Our results are more general in two ways: we give algorithms for more problems, including the aforementioned ones (see Tables 1 and 2), and on much broader digraph classes (see Figure 1). Naturally, the specific algorithms presented in [22] are more efficient than the algorithm following from our general framework. In the following, we discuss the above items in more detail.

Bi-mim-width. We introduce a new digraph width parameter, called *bi-mim-width*, which is a directed analogue of the mim-width of an undirected graph introduced by Vatshelle [44]. Roughly speaking, the bi-mim-width of a digraph G is defined as a branch-width with a cut

Table 1 Examples of $(\sigma^+, \sigma^-, \rho^+, \rho^-)$-sets, represented by finite or co-finite sets. For any row there is an associated NP-complete problem, usually maximizing or minimizing the cardinality of a set with the property. Some properties are known under different names; e.g. Efficient Total Dominating sets are also called Efficient Open Dominating sets, and here even the existence of such a set in a digraph G is NP-complete, as it corresponds to deciding if $V(G)$ can be partitioned by the open out-neighborhoods of some $S \subseteq V(G)$. If rows A and B have their in-restrictions and out-restrictions swapped for both σ and ρ (i.e. σ^+ of row A equals σ^- of row B and vice-versa, and same for ρ^+ and ρ^-), then a row-A set in G is always a row-B set in the digraph with all arcs of G reversed; this is the case for Dominating set vs in-Dominating set and for Kernel vs Independent Dominating set.

σ^+	σ^-	ρ^+	ρ^-	Standard name
$\{0\}$	$\{0\}$	$\mathbb{N} \setminus \{0\}$	$\mathbb{N}$	Kernel [45]
$\{0, ..., k-1\}$	$\{0\}$	$\{i : i \geq l\}$	$\mathbb{N}$	(k, l)-out Kernel [36]
$\mathbb{N}$	$\mathbb{N}$	$\mathbb{N}$	$\mathbb{N} \setminus \{0\}$	Dominating set [24]
$\{0\}$	$\{0\}$	$\mathbb{N}$	$\mathbb{N} \setminus \{0\}$	Independent Dominating set [16]
$\mathbb{N}$	$\mathbb{N}$	$\mathbb{N} \setminus \{0\}$	$\mathbb{N}$	In-Dominating set/Absorbing set [23]
$\mathbb{N}$	$\mathbb{N}$	$\mathbb{N} \setminus \{0\}$	$\mathbb{N} \setminus \{0\}$	Twin Dominating set [17]
$\mathbb{N}$	$\mathbb{N}$	$\mathbb{N}$	$\{i : i \geq k\}$	k-Dominating set [34]
$\mathbb{N}$	$\mathbb{N} \setminus \{0\}$	$\mathbb{N}$	$\mathbb{N} \setminus \{0\}$	Total Dominating set [2]
$\{0\}$	$\{0\}$	$\mathbb{N}$	$\{1\}$	Efficient (Closed) Dominating set [8]
$\mathbb{N}$	$\{1\}$	$\mathbb{N}$	$\{1\}$	Efficient Total Dominating set [37]
$\{k\}$	$\{k\}$	$\mathbb{N}$	$\mathbb{N}$	k-Regular Induced Subdigraph [15]

function that measures for a vertex partition (A, B) of G, the sum of the sizes of maximum induced matchings in two bipartite digraphs, one induced by edges from A to B, and the other induced by edges from B to A. This is similar to how rank-width is generalized to bi-rank-width for digraphs [29, 30]. We formally define bi-mim-width and linear bi-mim-width in Section 3. We compare bi-mim-width and other known width parameters. The mim-width of an undirected graph is exactly the half of the bi-mim-width of the digraph obtained by replacing each edge with bi-directed edges, and this observation can be used to argue that a bound on the bi-mim-width of a class of digraphs implies a bound on the mim-width of a certain class of undirected graphs.

Directed Locally Checkable Vertex (DLCV) Problems. We introduce directed locally checkable vertex subset (DLCVS) and partitioning (DLCVP) problems, in analogy with [43]. We abbreviate the union of these two families of problems to "DLCV problems". A DLCVS problem is represented as a $(\sigma^+, \sigma^-, \rho^+, \rho^-)$-problem for some $\sigma^+, \sigma^-, \rho^+, \rho^- \subseteq \mathbb{N}$, and it asks to find a maximum or minimum vertex set S in a digraph G such that for every vertex v in S, the numbers of out/in-neighbors in S are contained in σ^+ and σ^-, respectively, and for every vertex v in $V(G) \setminus S$, the numbers of out/in-neighbors in S are contained in ρ^+ and ρ^-, respectively. If each $\mu \in \{\sigma^+, \sigma^-, \rho^+, \rho^-\}$ is either finite or co-finite (i.e., $\mathbb{N} \setminus \mu$ is finite), then we say that the problem is *represented by finite or co-finite sets*. See Table 1 for several examples of DLCVS problems that appear in the literature and note that they are all represented by finite or co-finite sets. In particular, it includes the Kernel problem, which was introduced by von Neumann and Morgenstern [45].

A DLCVP problem is represented by a $(q \times q)$-matrix D for some positive integer q, where for all $i, j \in \{1, \ldots, q\}$, $D[i, j] = (\mu^+_{i,j}, \mu^-_{i,j})$ for some $\mu^+_{i,j}, \mu^-_{i,j} \subseteq \mathbb{N}$. The problem asks to find a vertex partition of a given digraph into $X_1, X_2, \ldots, X_q$ such that for all $i, j \in [q]$,

Table 2 Examples of directed LCVP problems that are represented by finite or co-finite sets. For every row there are choices of values for which the problems are NP-complete. For Directed H-Homomorphism let $V(H) = \{1, \ldots, |V(H)|\}$ and denote by $H\colon \overrightarrow{K_k}$ that H is an orientation of a complete graph on k vertices, and by $H\colon \overrightarrow{K_k^\circ}$ that H is an orientation of a complete graph on k vertices, with loops. (*) For Simple k-Coloring, we require two nonempty color classes to avoid trivial solutions. The general algorithm can easily be modified to take this into account.

Problem name	q	DLCVP $(q \times q)$-matrix D
Directed H-Homomorphism [25]	$\|V(H)\|$	$\forall (i,j) \in E(H)\colon D[i,j] = (\mathbb{N}, \mathbb{N})$ $\forall (i,j) \notin E(H)\colon D[i,j] = (\{0\}, \{0\})$
Oriented k-Coloring [18, 42]	k	$\bigvee_{H\colon \overrightarrow{K_k}}$ Directed H-Homomorphism
Simple k-Coloring (*) [40]	k	$\bigvee_{H\colon \overrightarrow{K_k^\circ}}$ Directed H-Homomorphism
$\exists$ $(\sigma^+, \sigma^-, \rho^+, \rho^-)$-set [This paper]	2	$\begin{pmatrix} (\sigma^+, \sigma^-) & (\mathbb{N}, \mathbb{N}) \\ (\rho^+, \rho^-) & (\mathbb{N}, \mathbb{N}) \end{pmatrix}$
$(\delta^+ \geq k_1, \delta^- \geq k_2)$-Partition [6]	2	$\begin{pmatrix} (\{j\colon j \geq k_1\}, \mathbb{N}) & (\mathbb{N}, \mathbb{N}) \\ (\mathbb{N}, \mathbb{N}) & (\mathbb{N}, \{j\colon j \geq k_2\}) \end{pmatrix}$
$(\delta^+ \geq k_1, \delta^+ \geq k_2)$-Partition [5]	2	$\begin{pmatrix} (\{j\colon j \geq k_1\}, \mathbb{N}) & (\mathbb{N}, \mathbb{N}) \\ (\mathbb{N}, \mathbb{N}) & (\{j\colon j \geq k_2\}, \mathbb{N}) \end{pmatrix}$
$(\Delta^+ \leq k_1, \Delta^+ \leq k_2)$-Partition [3]	2	$\begin{pmatrix} (\{j\colon j \leq k_1\}, \mathbb{N}) & (\mathbb{N}, \mathbb{N}) \\ (\mathbb{N}, \mathbb{N}) & (\{j\colon j \leq k_2\}, \mathbb{N}) \end{pmatrix}$
$(\delta^+ \geq k_1, \delta^- \geq k_2)$-Bipartite-Partition [4]	2	$\begin{pmatrix} (\mathbb{N}, \mathbb{N}) & (\{j\colon j \geq k_1\}, \mathbb{N}) \\ (\mathbb{N}, \{j\colon j \geq k_2\}) & (\mathbb{N}, \mathbb{N}) \end{pmatrix}$
$(\delta^+ \geq k_1, \delta^+ \geq k_2)$-Bipartite-Partition [4]	2	$\begin{pmatrix} (\mathbb{N}, \mathbb{N}) & (\{j\colon j \geq k_1\}, \mathbb{N}) \\ (\{j\colon j \geq k_2\}, \mathbb{N}) & (\mathbb{N}, \mathbb{N}) \end{pmatrix}$
2-Out-Coloring [1]	2	$\begin{pmatrix} (\mathbb{N} \setminus \{0\}, \mathbb{N}) & (\mathbb{N} \setminus \{0\}, \mathbb{N}) \\ (\mathbb{N} \setminus \{0\}, \mathbb{N}) & (\mathbb{N} \setminus \{0\}, \mathbb{N}) \end{pmatrix}$

the numbers of out/in-neighbors of a vertex of X_i in X_j are contained in $\mu^+_{i,j}$ and $\mu^-_{i,j}$, respectively. In analogy with subset problems, we say that the problem is *represented by finite or co-finite sets* if each set appearing in a pair that is an entry of D is either finite or co-finite. Directed H-Homomorphism is a directed LCVP problem represented by finite or co-finite sets: For a digraph H on vertices $\{1, \ldots, q\}$, we can view a homomorphism from a digraph G to H as a q-partition $(X_1, \ldots, X_q)$ of $V(G)$ such that we can only have an edge from X_i to X_j if the edge (i, j) is present in H. See Table 2. The Oriented k-Coloring problem, introduced by Sopena [41], asks whether there is a homomorphism to some orientation of a complete graph on at most k vertices, and can therefore be reduced to a series of directed LCVP problems. Removing the requirement that the color classes have to be independent sets, Smolíková [40] introduced the notion of a *simple k-coloring*, requiring however that the number of colors is at least two, to avoid trivial solutions. Several works in the literature concern problems of 2-partitioning the vertex sets of digraphs into parts with degree constraints either inside or between the parts of the partition [1, 3, 4, 5, 6]. All of these problems can be observed to be LCVP problems as well, see Table 2. Note that in the DLCVP-framework, we can consider q-partitions for any fixed $q \geq 2$, for all problems apart from 2-Out-Coloring. This fails for q-Out-Coloring, since this problem asks for a q-coloring with no monochromatic out-neighborhood.

▶ **Theorem 1.** *Directed LCVS and LCVP problems represented by finite or co-finite sets can be solved in time XP parameterized by bi-mim-width, when a branch decomposition is given.*

Furthermore, we show that the distance variants of DLCVS problems, for instance Distance-r Dominating Set can be solved in polynomial time on digraphs of bounded bi-mim-width. Another natural variant is the k-Kernel problem (see [7, Section 8.6.2]), which asks for a kernel in the $(k-1)$-th power of a given digraph. To show this, we prove that the r-th power of a digraph of bi-mim-width w has bi-mim-width at most rw (Lemma 12). For undirected graphs, there is a bound that does not depend on r [26], but we were not able to obtain such a bound for the directed case.

▶ **Theorem 2.** *Distance variants of directed LCVS problems represented by finite or co-finite sets can be solved in time XP parameterized by bi-mim-width, when a branch decomposition is given.*

Classes of intersection digraphs and their bi-mim-width. We provide various classes of digraphs of bounded bi-mim-width. We first summarize our results in the following theorem and give the background below. We illustrate the bounds in Figure 1.

▶ **Theorem 3.**

1. *Given a reflexive interval digraph, one can output a linear branch decomposition of bi-mim-width at most* 2 *in polynomial time. On the other hand, interval digraphs have unbounded bi-mim-width.*
2. *Given a representation of an adjusted permutation digraph G, one can construct in polynomial time a linear branch decomposition of G of bi-mim-width at most* 4*. Permutation digraphs have unbounded bi-mim-width.*
3. *Given a representation of an adjusted rooted directed path digraph G, one can construct in polynomial time a branch decomposition of G of bi-mim-width at most* 2*. Rooted directed path digraphs have unbounded bi-mim-width and adjusted rooted directed path digraphs have unbounded linear bi-mim-width.*
4. *Let H be an undirected graph. Given a representation of a reflexive H-digraph G, one can construct in polynomial time a linear branch decomposition of G of bi-mim-width at most* $12|E(H)|$*. P_2-digraphs, which are interval digraphs, have unbounded bi-mim-width.*
5. *Let H be an undirected graph. Given a nice H-convex digraph G with its bipartition (A, B), one can construct in polynomial time a linear branch decomposition of G of bi-mim-width at most* $12|E(H)|$*. P_2-convex digraphs have unbounded bi-mim-width.*
6. *Tournaments and directed acyclic graphs have unbounded bi-mim-width.*

(1. Interval digraphs) Recall that Müller [33] devised a recognition algorithm for interval digraphs, which also outputs a representation. By testing the reflexivity of a digraph, we can recognize reflexive interval digraphs, and output its representation. We convert it into a linear branch decomposition of bi-mim-width at most 2. On the other hand, interval digraphs generally have unbounded bi-mim-width. By Theorem 1, we can solve all DLCV problems on reflexive interval digraphs in polynomial time. This extends the polynomial-time algorithms for Independent Dominating Set and Kernel on interval nest digraphs given by Prisner [35], and includes polynomial-time algorithms for Absorbing Set, Dominating Set, and Kernel by Francis, Hell, and Jacob [22].

(2. Permutation digraphs) A *permutation digraph* is an intersection digraph of pairs of line segments whose endpoints lie on two parallel lines. Müller [33] considered permutation digraphs under the name “matching diagram digraph”, and observed that every interval digraph is a permutation digraph. Therefore, permutation digraphs have unbounded bi-mim-width. We say that a permutation digraph is *adjusted* if there exists one of the parallel lines, say Λ, such that for all $v \in V(G)$, S_v and T_v have the same endpoint in Λ. We show that every adjusted permutation digraph has linear mim-width at most 4.

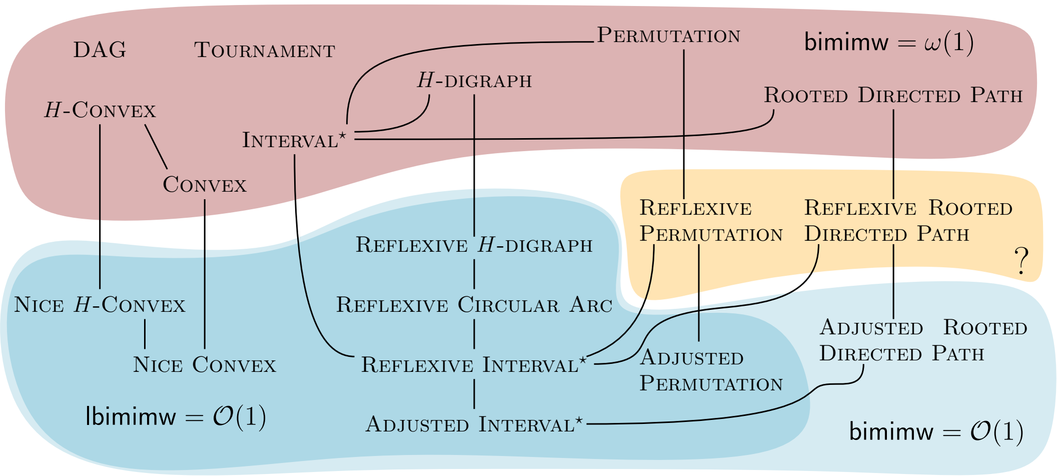


Figure 1 Digraph classes with bounds on their (linear) bi-mim-width. For graph classes marked with * there are polynomial-time algorithms to compute representations of their members. If digraph class A is depicted above B and there is an edge between A and B then $B \subseteq A$.

(3. Rooted directed path digraphs) It is known that chordal graphs have unbounded mim-width [28, 32]. As restrictions of chordal graphs, it has been shown that rooted directed path graphs, and more generally, leaf power graphs have mim-width at most 1 [26], while they have unbounded linear mim-width. A *rooted directed path digraph* is an intersection digraph of pairs of directed paths in a rooted directed tree (every node is reachable from the root), and it is *adjusted* if for every vertex v, the endpoint of S_v that is farther from the root is the same as the endpoint of T_v that is farther from the root. We show that every adjusted rooted directed path digraph has bi-mim-width at most 2. Since this class includes the biorientations of trees, it has unbounded linear bi-mim-width.

(4. H-digraphs) For an undirected graph H, an H*-graph* is an undirected intersection graph of connected subgraphs in an H-subdivision, introduced by Bíró, Hujter, and Tuza [11]. For example, interval graphs and circular-arc graphs are P_2-graphs and C_3-graphs, respectively. Fomin, Golovach, and Raymond [21] showed that H-graphs have linear mim-width at most $2|E(H)|+1$. Motivated by H-graphs, we introduce an H*-digraph* that is the intersection digraph of pairs of connected subgraphs in an H-subdivision (where H and its subdivision are undirected). We prove that reflexive H-digraphs have linear bi-mim-width at most $12|E(H)|$. This extends the linear bound of Fomin et al. [21] for H-graphs.

(5. H-convex digraphs) For an undirected graph H, a bipartite digraph G with bipartition (A, B) is an H*-convex digraph*, if there exists a subdivision F of H with $V(F) = A$ such that for every vertex b of B, each of the set of out-neighbors and the set of in-neighbors of v induces a connected subgraph in F. We say that an H-convex digraph is *nice* if for every vertex b of B, there is a bi-directed edge between b and some vertex of A. Note that H-convex graphs, introduced by Bonomo-Braberman et al. [12], can be seen as nice H-convex digraphs, by replacing every edge with bi-directed edges. We prove that nice H-convex digraphs have linear bi-mim-width at most $12|E(H)|$. This implies that H-convex graphs have linear mim-width at most $6|E(H)|$. For the special case when T is a tree with maximum degree Δ and t branching nodes, Bonomo-Braberman et al. [12] showed an improved bound of $2 + t(\Delta - 2)$ on the mim-width of T-convex graphs.

(6. Directed acyclic graphs and tournaments) We show that if H is the underlying undirected graph of a digraph G, then the bi-mim-width of G is at least the mim-width of H. Using this, we can show that acyclic orientations of grids have unbounded bi-mim-width. We also prove that tournaments have unbounded bi-mim-width. This refines an argument that they have unbounded bi-rank-width [7, Lemma 9.9.11].

We can summarize our algorithmic results as follows.

▶ **Corollary 4.** *Given a reflexive interval digraph, or a representation of either an adjusted permutation digraph, or an adjusted rooted directed path digraph, or a reflexive H-digraph, or a nice H-convex digraph, we can solve all DLCV problems represented by finite or co-finite sets, and their distance variants, in polynomial time.*

Related work. Intersection digraphs have first been considered by Beineke and Zamfirescu in 1982 [9]. Sen et al. [38] introduced the class of interval digraphs and Sen et al. [39] the class of circular-arc digraphs. Permutation digraphs were first studied under the name "matching diagram digraphs" by Müller [33]. Prisner [35] showed that the problems Clique, Chromatic Number, Independent Set, Partition into Cliques, Kernel, and Independent Dominating Set are polynomial-time solvable on interval *nest* digraphs, a subclass of interval digraphs G having a representation $\{(S_v, T_v) : v \in V(G)\}$ where for each vertex $v \in V(G)$, either $S_v \subseteq T_v$ or $T_v \subseteq S_v$. Very recently, and independently of this work, Francis, Hell, and Jacob [22] showed that Absorbing Set, Dominating Set, and Kernel are polynomial-time solvable on *reflexive* interval digraphs, a superclass of interval nest digraphs. They also showed that these problems remain hard on interval digraphs, even when all intervals are single points. Feder et al. [20] considered the List H-Homomorphism problem, but posing a structural restriction on H rather than the input graph. They showed that if H is an *adjusted* interval digraph, i.e. an interval digraph with a representation where both intervals associated with each vertex have the same left endpoint, then List H-Homomorphism is polynomial-time solvable.

The algorithmic result for undirected graphs analogous to ours is that all (undirected) locally checkable vertex problems are polynomial-time solvable if the input graph is given together with a decomposition of constant mim-width. This has been shown by Bui-Xuan, Telle, and Vatshelle [14]. In their work, the runtime of the algorithms is stated in terms of the number of equivalence classes of the *d-neighborhood equivalence* relation, and the connection between this notion and mim-width was made explicit by Belmonte and Vatshelle [10].

Organization of the paper. The paper is organized as follows. In Section 2, we introduce basic notations. In Section 3, we formally introduce bi-mim-width and compare with other known width parameters. In Section 4, we prove Theorem 3, and in Section 5, we prove Theorems 1 and 2. Proofs of statements marked with "⋆" are deferred to the full version.

2 Preliminaries

For a positive integer n, we use the shorthand $[n] := \{1, \ldots, n\}$.

Undirected Graphs. We use standard notions of graph theory and refer to [19] for an overview. All undirected graphs considered in this work are finite and simple. For a graph G, we denote by $V(G)$ the vertex set of G and $E(G)$ the edge set of G. For an edge $\{u, v\} \in E(G)$, we may use the shorthand "uv".

For two vertices $u, v \in V(G)$, the *distance* between u and v, denoted by $\text{dist}_G(u, v)$ or simply $\text{dist}(u, v)$, is the length of the shortest path between u and v. For $u \in V(G)$ and $A \subseteq V(G)$, we let $\text{dist}_G(u, A) = \min_{v \in A} \text{dist}_G(u, v)$.

Let G be a graph and $e = uv \in E(G)$. The *(edge) subdivision* of e is the operation of removing the edge e and adding a new vertex x and the edges ux and xv to G. A graph H is a *subdivision* of G if H can be obtained from G by a series of edge subdivisions. If H is a subdivision of G, then each vertex in $V(G)$ is called a *branching vertex* in H. A path P in H is called a *branching path* if its endpoints are branching vertices and no other vertices in P are branching vertices.

Digraphs. All digraphs considered in this work are finite and have no multiple edges, but may have loops. For a digraph G, we denote by $V(G)$ its vertex set and by $E(G) \subseteq V(G) \times V(G)$ its edge set. We say that an edge $(u, v) \in E(G)$ is directed from u to v. For a vertex v of G, we denote by $N_G^+(v)$ the set of out-neighbors of v, and by $N_G^-(v)$ the set of in-neighbors of v. If G is clear from the context, then we allow to remove G from the subscript.

A *rooted directed tree* is a digraph obtained from an undirected tree by selecting a root and directing all edges away from the root.

For a digraph G and two disjoint vertex sets $A, B \subseteq V(G)$, we denote by $G[A \to B]$ the bipartite digraph on bipartition (A, B) with edge set $E(G[A, B]) = E(G) \cap (A \times B)$, and denote by $G[A, B]$ the bipartite digraph on bipartition (A, B) with edge set $E(G[A \to B]) \cup E(G[B \to A])$. A set M of edges in a digraph G is a *matching* if no two edges share an endpoint, and it is an *induced matching* if there are no edges in G meeting two distinct edges in M. We denote by $\nu(G)$ the maximum size of an induced matching of G. For a vertex set A of G, we denote by $\overline{A} := V(G) \setminus A$. A vertex bipartition $(A, \overline{A})$ of G for some vertex set A of G is called a *cut*.

For two vertices $u, v \in V(G)$, the *distance* between u and v, denoted by $\text{dist}_G(u, v)$ or simply $\text{dist}(u, v)$, is the length of the shortest directed path from u to v. For a positive integer d, we denote by G^d the graph obtained from G by, for every pair (x, y) of vertices in G, adding an edge from x to y if there is a path of length at most d from x to y in G. We call it the *d-th power* of G.

3 Bi-mim-width

Throughout this section, definitions of concepts that are only touched on briefly can be found in Appendix A.

▶ **Definition 5** (Branch Decomposition). *Let Ω be a set. A* branch decomposition *over Ω is a pair $(T, \mathcal{L})$ of a subcubic tree T and a bijection $\mathcal{L}$ from Ω to the leaves of T. If T is a caterpillar, then $(T, \mathcal{L})$ is called a* linear branch decomposition *of G. For $e \in E(T)$, let T_A, T_B be the components of $T - e$. Let (A_e, B_e) be the cut of Ω where A_e is the set of elements that $\mathcal{L}$ maps to the leaves in T_A and B_e is the set of elements that $\mathcal{L}$ maps to the leaves in T_B.*

We introduce the bi-mim-width of a digraph. For a digraph G and $A \subseteq V(G)$, let $\text{mim}_G^+(A) := \nu(G[A \to \overline{A}])$, $\text{mim}_G^-(A) := \nu(G[\overline{A} \to A])$, and $\text{bimim}_G(A) := \text{mim}_G^+(A) + \text{mim}_G^-(A)$. A *branch decomposition of a digraph G* is a branch decomposition over $V(G)$.

▶ **Definition 6** (Bi-mim-width). *Let G be a digraph and $(T, \mathcal{L})$ be a branch decomposition of G. The* bi-mim-width *of $(T, \mathcal{L})$ is* $\text{bimimw}(T, \mathcal{L}) := \max_{e \in E(T)} (\text{bimim}_G(A_e))$. *The* bi-mim-width *of G, denoted by* $\text{bimimw}(G)$*, is the minimum bi-mim-width of any branch decomposition of G. The* linear bi-mim-width *of G, denoted by* $\text{lbimimw}(G)$*, is the minimum bi-mim-width of any linear branch decomposition of G.*

For an undirected graph G, we denote by $\mathrm{mimw}(G)$ its mim-width and by $\mathrm{lmimw}(G)$ its linear mim-width. The following two lemmas are clear by definition.

▶ **Lemma 7.** *Let G be a digraph and let H be an induced subdigraph of G. Then $\mathrm{bimimw}(H) \leq \mathrm{bimimw}(G)$ and $\mathrm{lbimimw}(H) \leq \mathrm{lbimimw}(G)$.*

▶ **Lemma 8.** *Let G be an undirected graph and let H be the biorientation of G. Then $\mathrm{mimw}(G) = \frac{\mathrm{bimimw}(H)}{2}$.*

We show that if a digraph G has small bi-mim-width, then its underlying undirected graph has small mim-width. But the other direction does not hold; the class of tournaments has unbounded bi-mim-width. We also argue that directed tree-width [27] and bi-mim-width are incomparable.

▶ **Lemma 9** (⋆)**.** *Let G be a digraph and let H be the underlying undirected graph of G. Then $\mathrm{mimw}(H) \leq \mathrm{bimimw}(G)$ and $\mathrm{lmimw}(H) \leq \mathrm{lbimimw}(G)$. On the other hand, the class of tournaments has unbounded bi-mim-width, while their underlying undirected graphs have linear mim-width 1.*

▶ **Lemma 10** (⋆)**.** *Directed tree-width and bi-mim-width are incomparable.*

We compare the bi-mim-width with the bi-rank-width of a digraph, introduced by Kanté [29]. Kanté and Rao [30] later generalized this notion to edge-colored graphs. For a digraph G, we denote by $\mathrm{birw}(G)$ its bi-rank-width and by $\mathrm{lbirw}(G)$ its linear bi-rank-width. We can verify that for every digraph G, $\mathrm{bimimw}(G) \leq \mathrm{birw}(G)$. Interestingly, we can further show that for every positive integer r, the bi-mim-width of the r-th power of G is at most the bi-rank-width of G. This does not depend on the value of r.

▶ **Lemma 11** (⋆)**.** *Let r and w be positive integers. If $(T, \mathcal{L})$ is a branch-decomposition of a digraph G of bi-rank-width w, then it is a branch-decomposition of G^r of bi-mim-width at most w.*

Next, we show that the r-th power of a digraph of bi-mim-width w has bi-mim-width at most rw. This will be used to prove Theorem 2.

▶ **Lemma 12.** *Let r and w be positive integers. If $(T, \mathcal{L})$ is branch-decomposition of a digraph G of bi-mim-width w, then it is a branch-decomposition of G^r of bi-mim-width at most rw.*

Proof. It is sufficient to prove that for every ordered vertex partition (A, B) of G, we have $\nu(G^r[A \to B]) \leq r\nu(G[A \to B])$. Assume $\nu(G[A \to B]) = t$ and suppose for contradiction that $\nu(G^r[A \to B]) \geq rt + 1$.

Let $\{(a_i, b_i) : i \in [rt{+}1]\}$ be an induced matching of $G^r[A \to B]$ with $\{a_i : i \in [rt{+}1]\} \subseteq A$. For each $i \in [rt + 1]$, let P_i be a directed path of length at most r from a_i to b_i in G. We choose an edge (c_i, d_i) in each P_i where $c_i \in A$ and $d_i \in B$. For each $i \in [rt + 1]$, let ℓ_i be the length of the subpath of P_i from a_i to c_i. Observe that $0 \leq \ell_i \leq r - 1$.

By the pigeonhole principle, there exists a subset I of $[rt + 1]$ of size at least $t + 1$ such that for all $i_1, i_2 \in I$, $\ell_{i_1} = \ell_{i_2}$. Since $\nu(G[A \to B]) = t$, there exist distinct integers $i_1, i_2 \in I$ such that there is an edge from c_{i_1} to d_{i_2}. Then there is a path of length at most d from a_{i_1} to b_{i_2}, contradicting the assumption that there is no edge from a_{i_1} to b_{i_2} in G^r. ◀

4 Classes of digraphs of bounded bi-mim-width

In this section we present several digraph classes of bounded bi-mim-width. Recall that a digraph G is an *intersection digraph* if there is a family of ordered pairs of sets $\{(S_v, T_v) \colon v \in V(G)\}$, called a *representation* of G, such that $(u, v) \in E(G)$ if and only if $S_u \cap T_v \neq \emptyset$. G

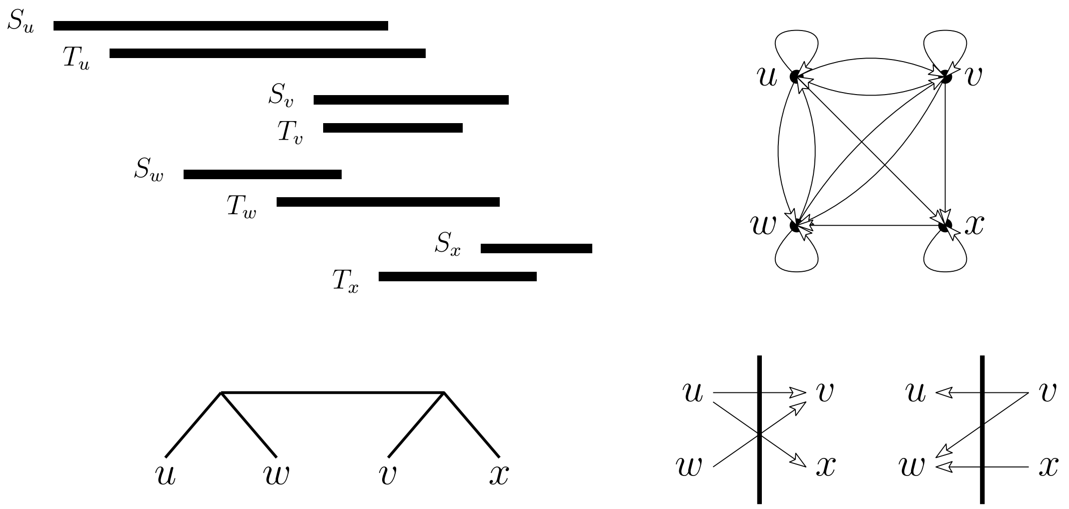


Figure 2 An example of a reflexive interval digraph. On the top left is its representation, on the top right one of its drawings, on the bottom left a linear branch decomposition and the bottom right shows that the cut associated with the "middle" edge of the branch decomposition has bi-mim-width value 2.

is called *reflexive* if for each $v \in V(G)$, $S_v \cap T_v \neq \emptyset$. Let H be a fixed undirected graph. A digraph G is an *H-digraph* if there is a subdivision F of H such that G is an intersection digraph of pairs of vertex sets inducing connected components in F.

▶ **Proposition 13.** *Let H be an undirected graph. Given a representation of a reflexive H-digraph G, one can construct in polynomial time a linear branch decomposition of G of bi-mim-width at most* $12|E(H)|$.

Proof. Let $m := |E(H)|$. We may assume that H is connected. If H has no edge, then it is trivial. Thus, we may assume that $m \geq 1$. Let G be a reflexive H-digraph, let F be a subdivision of H, and let $\mathcal{M} := \{(S_v, T_v) : v \in V(G)\}$ be a given reflexive H-digraph representation of G with underlying graph F. For each $v \in V(G)$, choose a vertex α_v in $S_v \cap T_v$. We may assume that vertices in $(\alpha_v : v \in V(G))$ are pairwise distinct and they are not branching vertices, by subdividing F more and changing $\mathcal{M}$ accordingly, if necessary.

We may assume that F has a branching vertex r, and we obtain a BFS ordering of F starting from r. We denote by $v <_B w$ if v appears before w in the BFS ordering. We give a linear ordering L of G such that for all $v, w \in V(G)$, if $\alpha_v <_B \alpha_w$, then v appears before w in L. This can be done in linear time. We claim that L has width at most $12m$. We choose a vertex v of G arbitrarily, and let A be the set of vertices in G that are v or a vertex appearing before v in L, and let $B := V(G) \setminus A$. It suffices to show $\text{bimim}_G(A) \leq 12m$. Let A^* be the set of vertices of F that are α_v or a vertex appearing before α_v, and let $B^* := V(F) \setminus A^*$. Let $\mathcal{P}$ be the set of paths in F such that

- for every $P \in \mathcal{P}$, P is a subpath of some branching path of F and it is a maximal path contained in one of A^* and B^*,
- $\bigcup_{P \in \mathcal{P}} V(P) = V(F)$.

Because of the property of a BFS ordering, it is easy to see that each branching path of F is partitioned into at most 3 vertex-disjoint paths in $\mathcal{P}$. Thus, we have $|\mathcal{P}| \leq 3m$. Note that two paths in $\mathcal{P}$ from two distinct branching paths may share an endpoint.

We first show that $\operatorname{mim}^+_G(A) \le 6m$. Suppose for contradiction that $G[A \to B]$ contains an induced matching M of size $6m+1$. By the pigeonhole principle, there is a subset $M_1 = \{(x_i, y_i) : i \in [3]\}$ of M of size 3 and a path P in $\mathcal{P}$ such that for every $(x, y) \in M_1$, S_x and T_y meet on P. Let p_1, p_2 be the endpoints of P.

Observe that $V(P) \subseteq A^*$ or $V(P) \subseteq B^*$. So, for each $i \in [3]$, it is not possible that α_{x_i} and α_{y_i} are both contained in $V(P)$. It implies that each connected component of $(S_{x_i} \cup T_{y_i}) \cap P$ contains an endpoint of P, as $S_{x_i} \cup T_{y_i}$ is connected. Therefore, there are at least two integers $j_1, j_2 \in [3]$ and a connected component C_1 of $(S_{x_{j_1}} \cup T_{y_{j_1}}) \cap P$ and a connected component C_2 of $(S_{x_{j_2}} \cup T_{y_{j_2}}) \cap P$ so that (1) C_1 and C_2 contain the same endpoint of P, and (2) for each $i \in [2]$, C_i contains a vertex of $S_{x_{j_i}}$ and a vertex of $T_{y_{j_i}}$. However, it implies that (x_{j_1}, y_{j_2}) or (x_{j_2}, y_{j_1}) is an edge, a contradiction.

We deduce that $\operatorname{mim}^+_G(A) \le 6m$. By a symmetric argument, we get $\operatorname{mim}^-_G(A) \le 6m$. Therefore, we have $\operatorname{bimim}_G(A) \le 12m$, as required. ◀

Interval digraphs are intersection digraphs of pairs of intervals over the real line, or, equivalently, P_2-digraphs. We first obtain a bound on the bi-mim-width of reflexive interval digraphs that improves the bound due to Proposition 13.

▶ **Proposition 14** (⋆). *Given a reflexive interval digraph, one can output a linear branch decomposition of bi-mim-width at most* 2 *in polynomial time.*

▶ **Proposition 15** (⋆). *Interval digraphs have unbounded bi-mim-width.*

A *permutation digraph* is an intersection digraph of pairs of line segments whose endpoints lie on two parallel lines. A permutation digraph G with representation $\{(S_v, T_v) \colon v \in V(G)\}$ is *adjusted* if for one of the two parallel lines, say Λ, it holds that all for all $v \in V(G)$, S_v and T_v have the same endpoint on Λ. We show that adjusted permutation digraphs have linear bi-mim-width at most 4.

▶ **Proposition 16** (⋆). *Given a representation of an adjusted permutation digraph G, one can construct in polynomial time a linear branch decomposition of G of bi-mim-width at most* 4.

Proof Sketch. Let $\Lambda_1 := \{(x, 0) : x \in \mathbb{R}\}$ and $\Lambda_2 := \{(x, 1) : x \in \mathbb{R}\}$ be two lines. Let G be a given adjusted permutation digraph with its representation $\{(S_v, T_v) : v \in V(G)\}$ where S_v and T_v are line segments whose endpoints lie on Λ_1 and Λ_2 and they have a common endpoint in Λ_1, say $(\alpha_v, 0)$. For each $v \in V(G)$, let $(\beta_v, 1)$ be the endpoint of S_v in Λ_2 and $(\gamma_v, 1)$ be the endpoint of T_v in Λ_2. We give a linear ordering L of G such that for all $v, w \in V(G)$, if $\alpha_v < \alpha_w$, then v appears before w in L.

We claim that L has bi-mim-width at most 4. We choose a vertex v of G arbitrarily, and let A be the set of vertices in G that are v or a vertex appearing before v in L, and let $B := V(G) \setminus A$. We verify that $\operatorname{mim}^+_G(A) \le 2$. By a symmetric argument, we have $\operatorname{mim}^-_G(A) \le 2$. Suppose for contradiction that $G[A \to B]$ has an induced matching $\{(v_i, w_i) : i \in [3]\}$ with $v_1, v_2, v_3 \in A$. Without loss of generality, we assume that $\alpha_{v_1} \le \alpha_{v_2} \le \alpha_{v_3}$. Observe that $\alpha_{w_1}, \alpha_{w_2} > \alpha_{v_3}$ and $\alpha_{w_3} \ge \alpha_{v_3}$. Let $w \in \{w_i : i \in [3]\}$ such that $|\alpha_w - \alpha_{v_3}|$ is minimum.

If $\alpha_w = \alpha_{v_3}$ and $\gamma_{w_3} > \beta_{v_3}$, then it is not difficult to verify that $\beta_{v_3} < \beta_{v_1}, \beta_{v_2}, \gamma_{w_1}, \gamma_{w_2} < \gamma_{w_3}$, as $\{(v_i, w_i) : i \in [3]\}$ is an induced matching. If $\beta_{v_1} \le \beta_{v_2}$, then T_{w_1} has to meet S_{v_2}, a contradiction. We can deal with other cases similarly. ◀

A *rooted directed path digraph* is an intersection digraph of pairs of directed paths in a rooted directed tree, and it is *adjusted* if for every vertex v, the endpoint of S_v that is farther from the root is the same as the endpoint of T_v that is farther from the root. We prove that adjusted rooted directed path digraphs have bounded bi-mim-width. We obtain a desired branch decomposition by attaching a leaf node corresponding to a vertex v to the common endpoint of S_v and T_v that is farther from the root, after finding an equivalent representation where the underlying directed tree has out-degree at most 2 and there are no two vertices v and w for which S_v and S_w share the endpoint farther from the root.

▶ **Proposition 17** ($\star$). *Given a representation of an adjusted rooted directed path digraph G, one can construct in polynomial time a branch decomposition of G of bi-mim-width at most 2. Adjusted rooted directed path digraphs have unbounded linear bi-mim-width.*

For an undirected graph H, a bipartite digraph G with bipartition (A, B) is an *H-convex digraph* if there is a subdivision F of H with $V(F) = A$ such that for every $b \in B$, both the set of out-neighbors of b and the set of in-neighbors of b induce a connected subgraph in F. An H-convex digraph is *nice* if every vertex of B is incident to some bi-directed edge. We prove that nice H-convex digraphs have linear bi-mim-width at most $12|E(H)|$. This proof resembles the proof of Proposition 13.

▶ **Proposition 18** ($\star$). *Let H be an undirected graph. Given a nice H-convex digraph G with its bipartition (A, B), one can construct in polynomial time a linear branch decomposition of G of bi-mim-width at most $12|E(H)|$.*

▶ **Proposition 19** ($\star$). *P_2-convex digraphs have unbounded bi-mim-width.*

5 Algorithmic applications

In this section we give the algorithmic applications of bi-mim-width. We show that all directed locally checkable vertex subset and all directed locally checkable vertex partitioning problems can be solved in XP time parameterized by the bi-mim-width of a given branch decomposition of the input digraph. We do so by adapting the framework of the d-neighborhood equivalence relation introduced by Bui-Xuan et al. [14] to digraphs.

d-Bi-neighhorhood-equivalence. The subsets of natural numbers that characterize locally checkable vertex subset/partitioning problems can be fully characterized when counting in- and out-neighbors up to some constant d. Therefore, if a vertex v has more than d for instance out-neighbors in two sets X and Y, then these two sets look the same to v in terms of its out-neighborhood. This is the main motivation for the following definition.

▶ **Definition 20.** *Let $d \in \mathbb{N}$. Let G be a digraph and $A \subseteq V(G)$. For two sets $X, Y \subseteq A$, we say that X and Y are d-*bi-neighbor* equivalent, written $X \equiv^{\pm}_{d,A} Y$, if* [1]

$$\forall u \in V(G) \setminus A\colon\ \min\{d, |N^-(u) \cap X|\} = \min\{d, |N^-(u) \cap Y|\} \text{ and}$$
$$\min\{d, |N^+(u) \cap X|\} = \min\{d, |N^+(u) \cap Y|\}.$$

We denote the number of equivalence classes of $\equiv^{\pm}_{d,A}$ by $\text{nec}(\equiv^{\pm}_{d,A})$. If $(T, \mathcal{L})$ is a branch decomposition of G, we let $\text{nec}_d(T, \mathcal{L}) = \max_{t \in V(T)} \max\{\text{nec}(\equiv^{\pm}_{d,V_t}), \text{nec}(\equiv^{\pm}_{d,\overline{V_t}})\}$.

The enumeration of equivalence classes is based on pairs of vectors called *d-bi-neighborhoods* of a subset X of A.

[1] Since the definition is given in terms of vertices from $\overline{A}$, we consider the directions of the edges in reverse, i.e., we consider $N^-(v)$ for $v \in \overline{A}$ when defining $\equiv^+$.

▶ **Definition 21.** *Let G be a digraph, $X \subseteq A \subseteq V(G)$, and $d \in \mathbb{N}$. The d-*out-neighborhood *of X, denoted by $U^+_{d,A}(X)$, and the d-*in-neighborhood *of X, denoted by $U^-_{d,A}(X)$ are the following vectors in $\{0,1,\ldots,d\}^{\overline{A}}$:*

$$U^+_{d,A}(X) = (\min\{d, |N^-(v) \cap X|\})_{v \in \overline{A}} \quad U^-_{d,A}(X) = (\min\{d, |N^+(v) \cap X|\})_{v \in \overline{A}}$$

*We refer to the pair $(U^+_{d,A}(X), U^-_{d,A}(X))$ as the d-*bi-neighborhood *$U^\pm_{d,A}(X)$; and we denote the set of all d-bi-neighborhoods as $\mathcal{U}^\pm_{d,A}$.*

There is a natural bijection between the d-bi-neighborhoods and the equivalence classes of $\equiv^\pm_{d,A}$.

▶ **Observation 22.** *Let G be a digraph and $X, Y \subseteq A \subseteq V(G)$. Then, $X \equiv^\pm_{d,A} Y$ if and only if $U^\pm_{d,A}(X) = U^\pm_{d,A}(Y)$.*

▶ **Lemma 23** ($\star$)**.** *Let G be a digraph on n vertices, $A \subseteq V(G)$, and $d \in \mathbb{N}$. There is an algorithm that enumerates all members of $\mathcal{U}^\pm_{d,A}$ in time $\mathcal{O}(\mathrm{nec}(\equiv^\pm_{d,A}) \log \mathrm{nec}(\equiv^\pm_{d,A}) \cdot dn^2)$. Furthermore, for each $Y \in \mathcal{U}^\pm_{d,A}$, the algorithm can provide some $X \subseteq A$ with $U^\pm_{d,A}(X) = Y$.*

5.1 Generalized Directed Domination Problems

The algorithm in this section is bottom-up dynamic programming along the given branch decomposition $(T, \mathcal{L})$ of the input digraph G, which we assume to be rooted in an arbitrary degree two node. For a node $t \in V(T)$, we let V_t be the vertices of G that are mapped to a leaf in the subtree of T rooted at t. We recall the formal definition of $(\sigma^+, \sigma^-, \rho^+, \rho^-)$-sets.

▶ **Definition 24.** *Let $\sigma^+, \sigma^-, \rho^+, \rho^- \subseteq \mathbb{N}$, and let $\Sigma = (\sigma^+, \sigma^-)$ and $\mathrm{R} = (\rho^+, \rho^-)$. Let G be a digraph and $S \subseteq V(G)$. We say that S $(\sigma^+, \sigma^-, \rho^+, \rho^-)$-dominates G, or simply that S (Σ, R)-dominates G, if:*

$$\forall v \in V(G)\colon |N^+(v) \cap S| \in \begin{cases} \sigma^+, & \text{if } v \in S \\ \rho^+, & \text{if } v \notin S \end{cases} \quad \text{and} \quad |N^-(v) \cap S| \in \begin{cases} \sigma^-, & \text{if } v \in S \\ \rho^-, & \text{if } v \notin S \end{cases}$$

▶ **Definition 25.** *Let $d(\mathbb{N}) = 0$. For a finite or co-finite set $\mu \subseteq \mathbb{N}$, let $d(\mu) = 1 + \min\{\max_{x \in \mathbb{N}} x \in \mu, \max_{x \in \mathbb{N}} x \notin \mu\}$. For finite or co-finite $\sigma^+, \sigma^-, \rho^+, \rho^- \subseteq \mathbb{N}$, $\Sigma = (\sigma^+, \sigma^-)$ and $\mathrm{R} = (\rho^+, \rho^-)$: $d(\sigma^+, \sigma^-, \rho^+, \rho^-) = d(\Sigma, \mathrm{R}) = \max\{d(\sigma^+), d(\sigma^-), d(\rho^+), d(\rho^-)\}$.*

As our algorithm progresses, it keeps track of partial solutions that may become a (Σ, R)-set once the computation has finished. This does not necessarily mean that at each node $t \in V(T)$, such a partial solution $X \subseteq V_t$ has to be a (Σ, R)-dominating set of $G[V_t]$. Instead, we additionally consider what is usually referred to as the "expectation from the outside" [14] in form of a subset Y of $\overline{V_t}$ such that $X \cup Y$ is a (Σ, R)-dominating set of $G[V_t]$.

▶ **Definition 26.** *Let $\mu^+, \mu^- \subseteq \mathbb{N}$ and let $\mathrm{M} = (\mu^+, \mu^-)$. Let G be a digraph, $A \subseteq V(G)$ and $X \subseteq V(G)$. We say that X M-dominates A if for all $v \in A$, we have that $|N^+(v) \cap X| \in \mu^+$ and $|N^-(v) \cap X| \in \mu^-$. Let Σ and R be as above. For $X \subseteq A$ and $Y \subseteq \overline{A}$, we say that (X, Y) (Σ, R)-dominates A, if $X \cup Y$ Σ-dominates X and $X \cup Y$ R-dominates $A \setminus X$.*

To describe an equivalence class $\mathcal{Q}$ of $\equiv^\pm_{d,A}$ we use the d-bi-neighbohoods of its members, which we denote by $\mathsf{desc}(\mathcal{Q})$. Note that by Observation 22, this is well-defined.

▶ **Definition 27.** *Let $\sigma^+, \sigma^-, \rho^+, \rho^- \subseteq \mathbb{N}$ be finite or co-finite, let $\Sigma = (\sigma^+, \sigma^-)$, $\mathrm{R} = (\rho^+, \rho^-)$, and $d = d(\Sigma, \mathrm{R})$. Let* opt *stand for* min *if we consider a minimization problem and for* max *if we consider a maximization problem. Let G be a digraph with branch decomposition $(T, \mathcal{L})$ and let $t \in V(T)$. For an equivalence class $\mathcal{Q}_t$ of $\equiv^{\pm}_{d,V_t}$, and an equivalence class $\mathcal{Q}_{\overline{t}}$ of $\equiv^{\pm}_{d,\overline{V_t}}$, we let:*

$$Tab_t[\mathsf{desc}(\mathcal{Q}_t), \mathsf{desc}(\mathcal{Q}_{\overline{t}})] = \begin{cases} \mathsf{opt}_{S \subseteq V_t} |S| \colon & S \in \mathcal{Q}_t \textit{ and for any } S_{\overline{t}} \in \mathcal{Q}_{\overline{t}}\colon \\ & (S, S_{\overline{t}})\ (\Sigma, \mathrm{R})\textit{-dominates } V_t \\ \infty & \textit{if } \mathsf{opt} = \min \textit{ and no such } S \textit{ exists} \\ -\infty & \textit{if } \mathsf{opt} = \max \textit{ and no such } S \textit{ exists} \end{cases}$$

We use the shorthand "$Tab_t[\mathcal{Q}_t, \mathcal{Q}_{\overline{t}}]$" for "$Tab_t[\mathsf{desc}(\mathcal{Q}_t), \mathsf{desc}(\mathcal{Q}_{\overline{t}})]$". For all such t, $\mathcal{Q}_t$, and $\mathcal{Q}_{\overline{t}}$, we initialize $Tab_t[\mathcal{Q}_t, \mathcal{Q}_{\overline{t}}]$ to be $-\infty$ if $\mathsf{opt} = \max$ and ∞ if $\mathsf{opt} = \min$.

Leaves of T. For a leaf $\ell \in V(T)$, let $v \in V(G)$ be such that $\mathcal{L}(v) = \ell$. Clearly, $\equiv_{d,\{v\}}$ has only two equivalence classes, namely the one containing $\emptyset$ and the one containing $\{v\}$. For each equivalence class $\mathcal{Q}$ of $\equiv_{d,V(G)\setminus\{v\}}$, let $R \in \mathcal{Q}$ which we can assume is given to us by Lemma 23. If $|N^+(v) \cap R| \in \sigma^+$ and $|N^-(v) \cap R| \in \sigma^-$, then $Tab_\ell[\{\{v\}\}, \mathcal{Q}] = 1$. If $|N^+(v) \cap R| \in \rho^+$ and $|N^-(v) \cap R| \in \rho^-$, then $Tab_\ell[\{\emptyset\}, \mathcal{Q}] = 0$.

Internal nodes of T. Let $t \in V(T)$ be an internal node with children a and b.

1. Consider each triple $\mathcal{Q}_a, \mathcal{Q}_b, \mathcal{Q}_{\overline{t}}$ of equivalence classes of $\equiv^{\pm}_{d,V_a}$, $\equiv^{\pm}_{d,V_b}$, and $\equiv^{\pm}_{d,\overline{V_t}}$, respectively.
2. Let $R_a \in \mathcal{Q}_a$, $R_b \in \mathcal{Q}_b$, and $R_{\overline{t}} \in \mathcal{Q}_{\overline{t}}$. Determine:
   - $\mathcal{Q}_{\overline{a}}$, the equivalence class of $\equiv^{\pm}_{d,\overline{V_a}}$ containing $R_b \cup R_{\overline{t}}$.
   - $\mathcal{Q}_{\overline{b}}$, the equivalence class of $\equiv^{\pm}_{d,\overline{V_b}}$ containing $R_a \cup R_{\overline{t}}$.
   - $\mathcal{Q}_t$, the equivalence class of $\equiv^{\pm}_{d,V_t}$ containing $R_a \cup R_b$.
3. Update $Tab_t[\mathcal{Q}_t, \mathcal{Q}_{\overline{t}}] = \mathsf{opt}\{Tab_t[\mathcal{Q}_t, \mathcal{Q}_{\overline{t}}], Tab_a[\mathcal{Q}_a, \mathcal{Q}_{\overline{a}}] + Tab_b[\mathcal{Q}_b, \mathcal{Q}_{\overline{b}}]\}$.

▶ **Theorem 28** (⋆)**.** *Let $\sigma^+, \sigma^-, \rho^+, \rho^- \subseteq \mathbb{N}$ be finite or co-finite, $\Sigma = (\sigma^+, \sigma^-)$, $\mathrm{R} = (\rho^+, \rho^-)$, and $d = d(\Sigma, \mathrm{R})$. There is an algorithm that given a digraph G on n vertices together with one of its branch decompositions $(T, \mathcal{L})$, computes and optimum-size (Σ, R)-dominating set in time $\mathcal{O}(\mathrm{nec}_d(T, \mathcal{L})^3 \cdot n^3 \log n)$. For $n \le \mathrm{nec}_d(T, \mathcal{L})$, the algorithm runs in time $\mathcal{O}(\mathrm{nec}_d(T, \mathcal{L})^3 \cdot n^2)$.*

▶ **Observation 29** (⋆)**.** *For $d \in \mathbb{N}$, a digraph G, and $A \subseteq V(G)$: $\mathrm{nec}(\equiv^{\pm}_{d,A}) \le n^{d \cdot \mathrm{bimim}_G(A)}$.*

▶ **Corollary 30.** *Let $\sigma^+, \sigma^-, \rho^+, \rho^- \subseteq \mathbb{N}$ be finite or co-finite, $\Sigma = (\sigma^+, \sigma^-)$, $\mathrm{R} = (\rho^+, \rho^-)$, and $d = d(\Sigma, \mathrm{R})$. Let G be a digraph on n vertices with branch decomposition $(T, \mathcal{L})$ of bi-mim-width $w \ge 1$. There is an algorithm that given any such G and $(T, \mathcal{L})$ computes an optimum-size (Σ, R)-dominating set in time $\mathcal{O}(n^{3dw+2})$.*

5.2 Directed Vertex Partitioning Problems

We now show that the locally checkable vertex partitioning problems can be solved in XP time parameterized by the bi-mim-width of a given branch decomposition. In analogy with [14], we lift the d-bi-neighborhood equivalence to q-tuples over vertex sets, which allows for devising the desired dynamic programming algorithm. The resulting algorithm follows a very similar strategy to the one for (Σ, R)-problems; the details are deferred to the full version (⋆).

▶ **Definition 31.** *A* bi-neighborhood-constraint *matrix is a $(q \times q)$-matrix D_q over pairs of finite or co-finite sets of natural numbers. Let G be a digraph, and $\mathcal{X} = (X_1, \ldots, X_q)$ be a q-partition of $V(G)$. We say that $\mathcal{X}$ is a D*-partition *if for all $i, j \in \{1, \ldots, q\}$ with $D_q[i, j] = (\mu^+_{i,j}, \mu^-_{i,j})$, we have that for all $v \in X_i$, $|N^+(v) \cap X_j| \in \mu^+_{i,j}$ and $|N^-(v) \cap X_j| \in \mu^-_{i,j}$. The* d-value *of D_q is $d(D_q) = \max_{i,j}\{d(\mu^+_{i,j}), d(\mu^-_{i,j})\}$.*

▶ **Theorem 32** (⋆). *Let D_q be a bi-neighborhood constraint matrix with $d = d(D_q)$. There is an algorithm that given a digraph G on n vertices together with one of its branch decompositions $(T, \mathcal{L})$, determines whether G has a D_q-partition in time $\mathcal{O}(\mathrm{nec}_d(T, \mathcal{L})^{3q} \cdot q \cdot n^3 \log n)$. For $n \leq \mathrm{nec}_d(T, \mathcal{L})$, the algorithm runs in time $\mathcal{O}(\mathrm{nec}_d(T, \mathcal{L})^{3q} \cdot q \cdot n^2)$.*

▶ **Corollary 33.** *Let D_q be a bi-neighborhood constraint matrix with $d = d(D_q)$. Let G be a digraph on n vertices with branch decomposition $(T, \mathcal{L})$ of bi-mim-width $w \geq 1$. There is an algorithm that given any such G and $(T, \mathcal{L})$ decides whether G has a D_q-partition in time $\mathcal{O}(q \cdot n^{3qdw+2})$.*

6 Conclusion

We introduced the digraph width measure bi-mim-width, and showed that (finitely represented) directed locally checkable vertex problems and their distance-r versions can be solved in polynomial time if the input digraph is given together with a branch decomposition of constant bi-mim-width. A natural next step in the understanding of this new parameter would be to determine the complexity of the Directed Feedback Vertex Set problem on digraphs of bounded bi-mim-width. We showed that several classes of intersection digraphs have constant bi-mim-width which adds a large number of polynomial-time algorithms for locally checkable problems related to domination and independence (given a representation) to the relatively sparse literature on the subject.

Intersection digraph classes such as interval digraphs seem too complex to give polynomial-time algorithms for optimization problems. Our work points to reflexivity as a reasonable additional restriction to give successful algorithmic applications of intersection digraphs, while maintaining a high degree of generality. This was observed independently for interval digraphs by Francis, Hell, and Jacob [22] who studied the Kernel, Absorbing Set, and Dominating Set problems. Apart from giving polynomial-time algorithms for these problems on reflexive interval digraphs, they showed that even for the severely restricted case when the intervals associated with the vertices are single points, the aforementioned problems remain hard.

Reflexivity presents a natural tractability barrier in the case of interval digraphs, or, more generally, H-digraphs for fixed H. The situation is not as clear yet when considering permutation digraphs or rooted directed path digraphs. Both digraph classes contain interval digraphs, therefore the hardness results from [22] apply as well. However, there are no matching polynomial-time algorithms for directed locally checkable vertex problems on reflexive permutation digraphs or reflexive rooted directed path digraphs; in particular, it is not known whether their bi-mim-width is bounded or not. We did show bounds on the bi-mim-width of their *adjusted* subclasses where we additionally require that every pair of objects representing a vertex share a common “endpoint” (where the concrete notion of endpoint depends on the considered type of representation). Arguably, reflexivity is the more natural restriction and one would hope that also in the case of these two digraph classes, it is the right barrier separating the tractable cases from the intractable ones. However, this question remains open for the time being.

References

1 Noga Alon, Jørgen Bang-Jensen, and Stéphane Bessy. Out-colourings of digraphs. *J. Graph Theory*, 93(1):88–112, 2020. doi:10.1002/jgt.22476.

2 S. Arumugam, K. Jacob, and Lutz Volkmann. Total and connected domination in digraphs. *Australas. J Comb.*, 39:283–292, 2007. URL: http://ajc.maths.uq.edu.au/pdf/39/ajc_v39_p283.pdf.

3 Jørgen Bang-Jensen, Stéphane Bessy, Frédéric Havet, and Anders Yeo. Out-degree reducing partitions of digraphs. *Theor. Comput. Sci.*, 719:64–72, 2018. doi:10.1016/j.tcs.2017.11.007.

4 Jørgen Bang-Jensen, Stéphane Bessy, Frédéric Havet, and Anders Yeo. Bipartite spanning sub(di)graphs induced by 2-partitions. *J. Graph Theory*, 92(2):130–151, 2019. doi:10.1002/jgt.22444.

5 Jørgen Bang-Jensen and Tilde My Christiansen. Degree constrained 2-partitions of semicomplete digraphs. *Theor. Comput. Sci.*, 746:112–123, 2018. doi:10.1016/j.tcs.2018.06.028.

6 Jørgen Bang-Jensen, Nathann Cohen, and Frédéric Havet. Finding good 2-partitions of digraphs II. Enumerable properties. *Theor. Comput. Sci.*, 640:1–19, 2016. doi:10.1016/j.tcs.2016.05.034.

7 Jørgen Bang-Jensen and Gregory Gutin, editors. *Classes of directed graphs*. Springer Monographs in Mathematics. Springer, Cham, 2018. doi:10.1007/978-3-319-71840-8.

8 David W. Bange, Anthony E. Barkauskas, Linda H. Host, and Lane H. Clark. Efficient domination of the orientations of a graph. *Discret. Math.*, 178(1-3):1–14, 1998. doi:10.1016/S0012-365X(97)81813-4.

9 Lowell W. Beineke and Christina M. Zamfirescu. Connection digraphs and second-order line digraphs. *Discrete Math.*, 39(3):237–254, 1982. doi:10.1016/0012-365X(82)90147-9.

10 Rémy Belmonte and Martin Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theor. Comput. Sci.*, 511:54–65, 2013. doi:10.1016/j.tcs.2013.01.011.

11 Miklós Biró, Mihály Hujter, and Zsolt Tuza. Precoloring extension. I. Interval graphs. *Discret. Math.*, 100(1-3):267–279, 1992. doi:10.1016/0012-365X(92)90646-W.

12 Flavia Bonomo-Braberman, Nick Brettell, Andrea Munaro, and Daniël Paulusma. Solving problems on generalized convex graphs via mim-width. In Anna Lubiw and Mohammad R. Salavatipour, editors, *Proceedings of the 17th International Symposium on Algorithms and Data Structures (WADS 2021)*, volume 12808 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2021. doi:10.1007/978-3-030-83508-8_15.

13 Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999. doi:10.1137/1.9780898719796.

14 Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013.

15 Domingos Moreira Cardoso, Marcin Kaminski, and Vadim V. Lozin. Maximum k-regular induced subgraphs. *J. Comb. Optim.*, 14(4):455–463, 2007. doi:10.1007/s10878-007-9045-9.

16 Michael Cary, Jonathan Cary, and Savari Prabhu. Independent domination in directed graphs. *Communications in Combinatorics and Optimization*, 6(1):67–80, 2021. doi:10.22049/cco.2020.26845.1149.

17 Gary Chartrand, Peter Dankelmann, Michelle Schultz, and Henda C. Swart. Twin domination in digraphs. *Ars Comb.*, 67, 2003.

18 Bruno Courcelle. The monadic second order logic of graphs VI: on several representations of graphs by relational structures. *Discret. Appl. Math.*, 54(2-3):117–149, 1994. doi:10.1016/0166-218X(94)90019-1.

19 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fourth edition, 2010. doi:10.1007/978-3-642-14279-6.

20 Tomás Feder, Pavol Hell, Jing Huang, and Arash Rafiey. Interval graphs, adjusted interval digraphs, and reflexive list homomorphisms. *Discrete Appl. Math.*, 160(6):697–707, 2012. doi:10.1016/j.dam.2011.04.016.

21 Fedor V. Fomin, Petr A. Golovach, and Jean-Florent Raymond. On the tractability of optimization problems on H-graphs. *Algorithmica*, 82(9):2432–2473, 2020. doi:10.1007/s00453-020-00692-9.

22 Mathew C. Francis, Pavol Hell, and Dalu Jacob. On the kernel and related problems in interval digraphs. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *Proceedings of the 32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl. doi:10.4230/LIPIcs.ISAAC.2021.17.

23 Yumin Fu. Dominating set and converse dominating set of a directed graph. *The American Mathematical Monthly*, 75(8):861–863, 1968. URL: http://www.jstor.org/stable/2314337.

24 J. Ghoshal, Renu Laskar, and D. Pillone. Topics on domination in directed graphs. In Theresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater, editors, *Domination in Graphs: Advanced Topics*. Taylor & Francis, 2017.

25 Pavol Hell and Jaroslav Nesetril. *Graphs and homomorphisms*, volume 28 of *Oxford lecture series in mathematics and its applications*. Oxford University Press, 2004.

26 Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. Mim-width III. Graph powers and generalized distance domination problems. *Theoret. Comput. Sci.*, 796:216–236, 2019. doi:10.1016/j.tcs.2019.09.012.

27 Thor Johnson, Neil Robertson, P. D. Seymour, and Robin Thomas. Directed tree-width. *J. Combin. Theory Ser. B*, 82(1):138–154, 2001. doi:10.1006/jctb.2000.2031.

28 Dong Yeap Kang, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. A width parameter useful for chordal and co-comparability graphs. *Theoret. Comput. Sci.*, 704:1–17, 2017. doi:10.1016/j.tcs.2017.09.006.

29 Mamadou Moustapha Kanté. The rank-width of directed graphs. *preprint*, 2007. arXiv:0709.1433.

30 Mamadou Moustapha Kanté and Michael Rao. The rank-width of edge-coloured graphs. *Theory Comput. Syst.*, 52(4):599–644, 2013. doi:10.1007/s00224-012-9399-y.

31 Stephan Kreutzer and O-joung Kwon. Digraphs of bounded width. In Jørgen Bang-Jensen and Gregory Z. Gutin, editors, *Classes of Directed Graphs*, Springer Monographs in Mathematics, pages 405–466. Springer, 2018. doi:10.1007/978-3-319-71840-8_9.

32 Stefan Mengel. Lower bounds on the mim-width of some graph classes. *Discrete Appl. Math.*, 248:28–32, 2018. doi:10.1016/j.dam.2017.04.043.

33 Haiko Müller. Recognizing interval digraphs and interval bigraphs in polynomial time. *Discrete Appl. Math.*, 78(1-3):189–205, 1997. doi:10.1016/S0166-218X(97)00027-9.

34 Lyes Ouldrabah, Mostafa Blidia, and Ahmed Bouchou. On the k-domination number of digraphs. *J. Comb. Optim.*, 38(3):680–688, 2019. doi:10.1007/s10878-019-00405-1.

35 Erich Prisner. Algorithms for interval catch digraphs. *Discret. Appl. Math.*, 51(1-2):147–157, 1994. doi:10.1016/0166-218X(94)90104-X.

36 Amina Ramoul and Mostafa Blidia. A new generalization of kernels in digraphs. *Discret. Appl. Math.*, 217:673–684, 2017. doi:10.1016/j.dam.2016.09.048.

37 Oliver Schaudt. Efficient total domination in digraphs. *J. Discrete Algorithms*, 15:32–42, 2012. doi:10.1016/j.jda.2012.02.003.

38 M. Sen, S. Das, A. B. Roy, and D. B. West. Interval digraphs: an analogue of interval graphs. *J. Graph Theory*, 13(2):189–202, 1989. doi:10.1002/jgt.3190130206.

39 M. Sen, S. Das, and Douglas B. West. Circular-arc digraphs: a characterization. *J. Graph Theory*, 13(5):581–592, 1989. doi:10.1002/jgt.3190130508.

40 Petra Smolíková. The simple chromatic number of oriented graphs. *Electronic Notes in Discrete Mathematics*, 5:281–283, 2000. doi:10.1016/S1571-0653(05)80186-6.

41 Eric Sopena. The chromatic number of oriented graphs. *J. Graph Theory*, 25(3):191–205, 1997. doi:10.1002/(SICI)1097-0118(199707)25:3<191::AID-JGT3>3.0.CO;2-G.

42 Éric Sopena. Homomorphisms and colourings of oriented graphs: An updated survey. *Discret. Math.*, 339(7):1993–2005, 2016. doi:10.1016/j.disc.2015.03.018.

43 Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997. doi:10.1137/S0895480194275825.

44 Martin Vatshelle. *New Width Parameters of Graphs*. PhD thesis, Univ. Bergen, 2012.

45 John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, New Jersey, 1944.

A Additional Definitions

For an undirected graph G and two disjoint vertex sets $A, B \subseteq V(G)$, we denote by $G[A, B]$ the bipartite graph on bipartition (A, B) such that $E(G[A, B])$ is exactly the set of edges of G incident with both A and B.

For a graph G and a set $A \subseteq V(G)$, we let $\text{mim}_G(A) := \nu(G[A, \overline{A}])$. A *branch decomposition of* G is a branch decomposition over $V(G)$ (recall Definition 5).

▶ **Definition 34** (Mim-width). *Let G be a graph and $(T, \mathcal{L})$ be a branch decomposition of G. The* mim-width of $(T, \mathcal{L})$ *is* $\text{mimw}(T, \mathcal{L}) := \max_{e \in E(T)} \text{mim}_G(A_e)$. *The* mim-width of G, *denoted by* $\text{mimw}(G)$, *is the minimum mim-width over all branch decompositions of G. The* linear mim-width of G, *denoted by* $\text{lmimw}(G)$, *is the minimum mim-width over all linear branch decompositions of G.*

Let G be a digraph, and $A, B \subseteq V(G)$ be two disjoint vertex sets. We let $M_G[A \to B]$ the matrix whose columns are indexed by A and whose rows are indexed by B such that for $a \in A$ and $b \in B$, $M_G[A \to B](a, b) = 1$ if $(a, b) \in E(G)$ and $M_G[A \to B](a, b) = 0$ otherwise. For each $A \subseteq V(G)$,we let $\text{cutrk}^+_G(A) := \text{rank}(M_G[A \to \overline{A}])$ and $\text{cutrk}^-_G(A) := \text{rank}(M_G[\overline{A} \to A])$. We let $\text{bicutrk}_G(A) := \text{cutrk}^+_G(A) + \text{cutrk}^-_G(A)$.

▶ **Definition 35** (Bi-rank-width). *Let G be a digraph, and $(T, \mathcal{L})$ be a branch decomposition of G. The* bi-rank-width *of $(T, \mathcal{L})$ is* $\max_{e \in E(T)} \text{bicutrk}_G(A_e)$. *The (linear)* bi-rank-width *of G is the minimum bi-rank-width of any (linear) branch decomposition of G.*

Let T be a rooted directed tree. For a vertex $t \in V(T)$, we denote by T_t the subtree of T containing all vertices v such that there is a directed path from t to v in T.

▶ **Definition 36** (Strong guard). *Let G be a digraph and $X, Y \subseteq V(G)$. We say that Y* is a strong guard for X *if every walk starting and ending in X, and containing a vertex from $V(G) \setminus X$, contains a vertex from Y.*

▶ **Definition 37** (Directed treewidth). *Let G be a digraph. A* directed tree decomposition *is a triple (T, β, γ) of a rooted directed tree T and two maps $\beta\colon V(T) \to 2^{V(G)}$ and $\gamma\colon E(T) \to 2^{V(G)}$,*

1. *The set $\{\beta(t)\colon t \in V(T)\}$ is a partition of $V(G)$.*
2. *For each $e = (u, v) \in E(T)$, $\gamma(e)$ is a strong guard for $\bigcup_{t \in V(T_v)} \beta(t)$.*

For each $t \in V(T)$, we let $\Gamma(t) := \beta(t) \cup \bigcup_{e \sim t} \gamma(e)$, where $e \sim t$ means that e is incident with t. The width *of (T, β, γ) is $\max_{t \in V(T)} |\Gamma(t)| - 1$, and the* directed treewidth *of a digraph G is the minimum width over all its directed tree decompositions.*

We refer to [31] for an introduction to the width measures bi-rank-width and directed treewidth.

Further Exploiting c-Closure for FPT Algorithms and Kernels for Domination Problems

Lawqueen Kanesh
National University of Singapore, Singapore

Jayakrishnan Madathil
Chennai Mathematical Institute, India

Sanjukta Roy
Algorithms and Complexity Group, TU Wien, Austria

Abhishek Sahu
The Institute of Mathematical Sciences, HBNI, Chennai, India

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway

Abstract

For a positive integer c, a graph G is said to be c-closed if every pair of non-adjacent vertices in G have at most $c-1$ neighbours in common. The closure of a graph G, denoted by $cl(G)$, is the least positive integer c for which G is c-closed. The class of c-closed graphs was introduced by Fox et al. [ICALP '18 and SICOMP '20]. Koana et al. [ESA '20] started the study of using $cl(G)$ as an additional structural parameter to design kernels for problems that are W-hard under standard parameterizations. In particular, they studied problems such as Independent Set, Induced Matching, Irredundant Set and (Threshold) Dominating Set, and showed that each of these problems admits a polynomial kernel, either w.r.t. the parameter $k+c$ or w.r.t. the parameter k for each fixed value of c. Here, k is the solution size and $c = cl(G)$. The work of Koana et al. left several questions open, one of which was whether the Perfect Code problem admits a fixed-parameter tractable (FPT) algorithm and a polynomial kernel on c-closed graphs. In this paper, among other results, we answer this question in the affirmative. Inspired by the FPT algorithm for Perfect Code, we further explore two more domination problems on the graphs of bounded closure. The other problems that we study are Connected Dominating Set and Partial Dominating Set. We show that Perfect Code and Connected Dominating Set are fixed-parameter tractable w.r.t. the parameter $k + cl(G)$, whereas Partial Dominating Set, parameterized by k is W[1]-hard even when $cl(G) = 2$. We also show that for each fixed c, Perfect Code admits a polynomial kernel on the class of c-closed graphs. And we observe that Connected Dominating Set has no polynomial kernel even on 2-closed graphs, unless NP $\subseteq$ co-NP/poly.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases c-closed graphs, domination problems, perfect code, connected dominating set, fixed-parameter tractable, polynomial kernel

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.39

Funding *Lawqueen Kanesh*: Supported in part by NRF Fellowship for AI grant [R-252-000-B14-281] and by Defense Service Organization, Singapore.
Jayakrishnan Madathil: Supported by Chennai Mathematical Institute and the Infosys Foundation.
Sanjukta Roy: Supported by the WWTF research grant VRG18012.
Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

Acknowledgements We thank two anonymous reviewers for their suggestions that helped improve the presentation of this article.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 39; pp. 39:1–39:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

For a positive integer c, a graph G is said to be c-closed if every pair of non-adjacent vertices in G have at most $c-1$ neighbours in common. That is, for distinct vertices u and v of G, $|N(u) \cap N(v)| \leq c-1$ if $uv \notin E(G)$. In this paper, we investigate the parameterized complexity of domination problems on the class of c-closed graphs. The problems that we study are Perfect Code, Connected Dominating Set and Partial Dominating Set. All these problems are W[1]-hard (w.r.t. standard parameters) on general graphs [12, 18, 19], and their complexities on various restricted graph classes have been studied extensively [4, 15, 22, 29, 30, 31, 34, 41, 44].

Fox et al. [25, 26] introduced the class of c-closed graphs in 2018 as a "distribution-free" model of social networks. While the literature abounds with models that attempt to capture the structure of social networks, they are all probabilistic models. (See, for instance, the survey by Chakrabarti and Faloutsos [13].) And in an attempt to capture the spirit of "social-network-like" graphs without relying on probabilistic models, Fox et al. [26] "turn[ed] to one of the most agreed upon properties of social networks – triadic closure, the property that when two members of a social network have a friend in common, they are likely to be friends themselves." It is easy to see that the definition of c-closed graphs is a reasoned approximation of this property. In a c-closed graph, every pair of vertices with at least c common neighbours are adjacent to each other. Fox et al. [26, Table A.1], and later Koana et al. [39, Table 1], showed that several social networks and biological networks are indeed c-closed for rather small values of c.

Fox et al. [26] showed that an n-vertex c-closed graph contains at most $3^{c/3} \cdot n^2$ maximal cliques.[1] This bound, when coupled with an algorithm for enumerating all maximal cliques in a graph, yields a $2^{\mathcal{O}(c)} \cdot \text{poly}(n)$ time algorithm that enumerates all maximal cliques in c-closed graphs. Observe that an algorithm that *enumerates all maximal cliques* in a graph can be used to determine if a graph *contains a clique of a given size* as well. Thus, the Clique problem, which, given a graph G and an integer k as input, asks if G contains a clique of size k, is fixed-parameter tractable with respect to the parameter c. Notice that Clique, when parameterized by k, is W[1]-complete on general graphs [18], and therefore does not admit a fixed-parameter tractable algorithm unless FPT=W[1].

In light of this result, we could very well ask: How do other problems that are W-hard on general graphs fare on the class of c-closed graphs? In particular, is Independent Set, another canonical W[1]-complete problem [18], fixed-parameter tractable on c-closed graphs? Koana et al. [39] showed that Independent Set, which takes a graph G and an integer k as input, and asks if G contains an independent set of size k, is indeed fixed-parameter tractable w.r.t. the parameter $k+c$. In fact, by applying a "Buss-like" reduction rule [9], they showed that the problem admits a kernel with ck^2 vertices. Motivated by this example, they studied the (kernelization) complexity of three more problems – Induced Matching, Irredundant Set and Threshold Dominating Set (TDS) – and showed that these problems admit polynomial kernels (either w.r.t. the parameter $k+c$, or w.r.t. the parameter k for each fixed c.) TDS is a variant of Dominating Set in which each vertex needs to be dominated at least r times for a given integer r. The kernels for the first two of these problems have size poly(c,k) whereas the kernel for TDS has size $k^{\mathcal{O}(cr)}$. They also designed an FPT algorithm for TDS that runs in time $3^{c/3} + (ck)^{\mathcal{O}(rk)} n^{\mathcal{O}(1)}$. A key ingredient in all

[1] Note that the classic Moon-Moser theorem only guarantees an upper bound of $3^{n/3}$ for the number of maximal cliques in an n-vertex graph [47].

these results was a polynomial bound for the Ramsey number on c-closed graphs. Koana et al. [39] proved that every c-closed graph with $\mathcal{O}(cb^2 + ab)$ vertices contains either a clique of size a or an independent set of size b, and predicted that this bound could be useful in settling the parameterized complexity of other problems as well. In this paper, we use this bound, and show that two variants of Dominating Set admit fixed-parameter tractable algorithms on c-closed graphs. In particular, we show that Perfect Code is FPT on c-closed graphs, and thus settle a question left open by Koana et al. [39].

Closure of a graph. Recall that a graph G is said to be c-closed if every pair of non-adjacent vertices have at most $c-1$ neighbours in common. The *closure*[2] of a graph G, denoted by $cl(G)$, is the least positive integer c for which G is c-closed. Notice that $cl(G) = 1 + \max\{|N(u) \cap N(v)| \mid u, v \in V(G), uv \notin E(G)\}$, and therefore $cl(G)$ can be computed in polynomial time. In this paper, we study the parameterized complexity of some of the widely-studied problems on graphs of bounded closure, and thus attempt to present a more comprehensive answer to the following questions. How good a structural parameter is $cl(G)$ when it comes to the tractability of domination problems? And in this regard, how does $cl(G)$ differ from some of the other widely-studied structural parameters such as maximum degree, degeneracy and treewidth? Observe that if the maximum degree of graph G is $\Delta(G)$, then $cl(G) \leq \Delta(G) + 1$. But the comparability ends there. As noted by Koana et al. [39], an n-vertex clique is 1-closed, but has degeneracy and treewidth $n-1$. On the other hand, the complete bipartite graph $K_{2,n-2}$ has treewidth and degeneracy 2, but $cl(K_{2,n-2}) = n-1$. Thus, closure is incomparable with degeneracy and treewidth. We also note that when parameterized by $cl(G)$ alone, most of the widely-studied problems, with the exception of Clique, would be para-NP-hard. This applies to problems such as Vertex Cover, Independent Set, Dominating Set, Connected Dominating Set and Perfect Code, as all these problems are NP-hard on graphs of maximum degree 4 [21, 27], and therefore NP-hard on 5-closed graphs. So this parameter alone is too small to yield tractability results, and therefore, has to be used in combination with some other parameter, for example, the solution size. But this is often the case with other structural parameters such as degeneracy and maximum degree as well; they are often combined with the solution size [3, 48].

Our results and methods. Let us first define the concept of domination in graphs. Consider a graph G. We say that a vertex in G dominates itself and all its neighbours. That is, a vertex v dominates $N[v]$. And for a set $V' \subseteq V(G)$, V' dominates $N[V']$. A *dominating set* of a graph is a set of vertices $D \subseteq V(G)$ that dominates the entire vertex set, i.e., $N[D] = V(G)$. Or equivalently, $D \subseteq V(G)$ is a dominating set of G if $|D \cap N[v]| \geq 1$ for every vertex $v \in V(G)$. A dominating set $D \subseteq V(G)$ is said to be a *connected dominating set* of G if $G[D]$ is a connected subgraph of G. A *perfect code* of G is a dominating set of G that dominates every vertex exactly once. That is, $D \subseteq V(G)$ is a perfect code of G if $|D \cap N[v]| = 1$ for every vertex $v \in V(G)$. For a non-negative integer t, a set of vertices $V' \subseteq V(G)$ is said to be a *t-partial dominating set* of G if V' dominates at least t vertices of G, i.e., if $|N[V']| \geq t$.

[2] Koana et al. [39] use the term c-closure instead of closure. But we believe that closure is more appropriate. We must note that the term closure is already used in existing graph theory literature to refer to a certain super-graph of a graph [8, p. 486]. But for that matter, so is the term k-closure [7]. We believe that given the context, there is no room for ambiguity.

In the Perfect Code (resp. Connected Dominating Set (CDS)) problem, the input consists of an n-vertex graph G and a non-negative integer k, and the question is to decide if G contains a perfect code (resp. connected dominating set) of size at most k. In the Partial Dominating Set (PDS) problem, the input consists of an n-vertex graph G and two non-negative integers k and t, and the question is to decide if G contains a t-partial dominating set of size at most k. We show that Perfect Code and CDS, when parameterized by $k+cl(G)$, are fixed-parameter tractable, whereas PDS, when parameterized by k, is W[1]-hard, even for $cl(G) = 2$. Specifically, we prove the following results. (Here, $n = |V(G)|$ and $c = cl(G)$.)

1. Perfect Code admits a fixed-parameter tractable algorithm that runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$. Moreover, for each fixed $c \geq 1$, Perfect Code admits a kernel with $k^{\mathcal{O}(2^c)}$ vertices on the family of c-closed graphs.
2. CDS admits a fixed-parameter tractable algorithm that runs in time $2^{\mathcal{O}(ck^2\log(ck))}n^{\mathcal{O}(1)}$. But CDS does not admit a polynomial kernel when parameterized by k even when $cl(G) = 2$, unless NP $\subseteq$ co-NP/poly. (The kernelization lower bound follows from a result due to Misra et al. [44].)
3. PDS, when parameterized by k, is W[1]-hard on 2-closed graphs.

Note that a perfect code and a connected dominating set are both dominating sets. Naturally, our algorithms for Perfect Code and CDS rely on three crucial properties of dominating sets and c-closed graphs. Consider a c-closed graph G, and a dominating set D of G of size k. **(P1)** If G contains an independent set I of size $k+1$, then by the pigeonhole principle, there exists a vertex $v \in D$ that dominates at least two vertices of I. That is, $v \in N(u) \cap N(u')$ for a pair of vertices $u, u' \in I$ (Lemma 11). **(P2)** The dominating set D must intersect every "large" maximal clique (Corollary 7). This follows from the fact that any vertex outside a maximal clique can dominate at most $c-1$ vertices of the clique (Lemma 6). Thus, if G contains a maximal clique of size $(c-1)k+1$, say Q, then we must have $D \cap V(Q) \neq \emptyset$. **(P3)** If G contains ℓ distinct "large" maximal cliques, then G contains an independent set of size ℓ as well (Lemma 8). This again is a consequence of the property that any vertex outside a maximal clique has at most $c-1$ neighbours in the clique. Here, depending on each problem, we will define an appropriate lower bound on the size of a clique for it to be large. But in both the problems, this bound will be poly(c,k). Finally, we use the following two results due to Koana et al. [39]. **(R1)** Every c-closed graph with $\mathcal{O}(cb^2+ab)$ vertices contains either a clique of size a or an independent set of size b (Lemma 1). **(R2)** We can find a $(k+1)$-sized independent set of an n-vertex c-closed graph, if it exists, or correctly conclude that no such set exists, in time $2^{\mathcal{O}(k\log(ck))}n^{\mathcal{O}(1)}$ (Corollary 4).

We now briefly outline how our algorithms exploit these properties. In light of (P1), we first find an independent set I of size $k+1$ using (R2), and branch on the vertices in $\bigcup_{u,u' \in I} N(u) \cap N(u')$. Note that since $|I| = k+1$, we have $\binom{k+1}{2} = \mathcal{O}(k^2)$ choices for the pair $\{u, u'\}$. And for each pair $u, u' \in I$, we have $|N(u) \cap N(u')| \leq c-1$ as G is c-closed. Once this branching step is exhaustively applied, every independent set in G has size at most k. But then (P3) will imply that G contains at most k "large" maximal cliques. Now we partition the vertex set of G into two parts, L and R, where L is the set of vertices that belong to at least one large maximal clique and R the set of remaining vertices. Thus, L is the union (not necessarily disjoint) of at most k large cliques. And the subgraph $G[R]$ contains no large clique or no independent set of size $k+1$. Therefore, by (R1), we will have $|R| = $ poly(c,k). So we can guess the set of vertices from R that belongs to the "dominating set" that we are looking for, in case (G,k) is indeed a yes-instance. And corresponding to each such guess, we then use the property that L is a union of cliques to solve the problem appropriately. For example, in the case of Perfect Code, we show that once we guess the subset of R that belongs to the solution, the problem then reduces to solving an instance of the d-Exact Hitting Set problem (a variant of Hitting Set in which every set has size at most d and needs to be hit exactly once) for an appropriate choice of d, which can then be solved in time $d^k n^{\mathcal{O}(1)}$. In the case of CDS, we reduce the final step to $2^{\text{poly}(c,k)}$ many instances of the (edge-weighted) Steiner Tree problem, a common technique used in algorithms that seek connected solutions [32, 44, 45, 46]. And we will have the guarantee that our original CDS instance is a yes-instance if and only if at least one of the Steiner Tree instances is a yes-instance. We prove the W-hardness of PDS by designing a parameterized reduction from the Independent Set problem on regular graphs, which is known to be W[1]-complete [10]. The inadmissibility of a polynomial kernel for CDS follows from a result due to Misra et al. [44], which says that CDS admits no polynomial kernel on graphs of girth 5, and the fact that graphs of girth 5 are 2-closed.

To design our kernel for Perfect Code, we bound the size of independent sets and cliques in the input graph by $k^{\mathcal{O}(2^c)}$, and then invoke (R1). The main ingredient in bounding the independent set size is a reduction rule, by which we find a sufficiently large independent set with sufficiently many common neighbours and delete an arbitrary vertex from that independent set. To find this independent set, we design an algorithm that works as follows: Given a c-closed graph G and an integer k, the algorithm will either output an independent set of size k or correctly report that every independent set in G has size $\text{poly}(c,k)$ (Lemma 10). After an exhaustive application of this reduction rule, every independent set in the input graph will have bounded size, and by (P3), the graph will contain only a bounded number of large cliques. Then, we bound the size of each clique as well, which, by (R1), will result in the kernel. We note that our fixed-parameter tractable algorithm and polynomial kernel for Perfect Code do not imply each other. The kernel runs in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(c)}$, and therefore, does not imply a fixed-parameter tractable algorithm w.r.t the parameter $k+c$.

We must point out that properties (P1) and (P2) have been used by Koana et al. [39] in their algorithm and kernel for the TDS problem. But these properties alone are inadequate for Perfect Code and CDS. Hence we introduce (P3), which bounds the number of large maximal cliques in terms of the maximum size of an independent set. We also note that while properties (P1) and (P2) are specific to domination problems, (P3) is a general-purpose bound. Our strategy of partitioning the vertices into L and R (vertices of large cliques and the remaining vertices) is also not specific to domination problems, and could be applicable to other problems as well. So is Lemma 10, which, as mentioned above, gives an algorithm that either outputs an independent set of size k or guarantees an upper bound of $\text{poly}(c,k)$ on the independent set size. We use Lemma 10 to fashion a reduction rule (Reduction Rule 19), which we use to bound the size of independent sets while designing our kernel for Perfect Code. The idea behind Reduction Rule 19 is as follows. To bound the size of any independent in the graph, it is sufficient to bound the size of independent sets within the induced subgraph $G[N(v)]$ for every $v \in V(G)$. Then, to bound the size of independent sets in $G[N(v)]$, it is sufficient to bound the size of independent sets in $G[N(v) \cap N(u)]$ for every $u \in V(G) \setminus \{v\}$. And to bound the size of independent sets in $G[N(v) \cap N(u)]$, it is sufficient to bound the size of independent sets in $G[N(v) \cap N(u) \cap N(w)]$ for every $w \in V(G) \setminus \{v, u\}$ and so on. This strategy of successively bounding the independent sets in stages could be applicable to other problems on c-closed graphs as well. Since G is c-closed, we only need to continue for $c-1$ stages. That is, we only need to bound the size of independent sets in $G[\cap_{x \in Y} N(x)]$ for all $Y \subseteq V(G)$ with $|Y| = c-1$.

Related work on domination problems. Domination problems have long been the subject of extensive research in algorithmic graph theory. All the domination problems discussed above are W-hard on general graphs, when parameterized by the solution size. Therefore, a great deal of effort has gone into studying the complexity of these problems on various graph classes. In particular, the classic Dominating Set problem is known to be W[2]-complete [19] on general graphs, and W[2]-hard even on bipartite graphs (and hence on triangle-free graphs) [49], but admits a fixed-parameter tractable algorithm on graphs of girth at least 5 [49], planar graphs [1, 2, 24, 35], graphs of bounded genus [20], map graphs [16], H-minor free graphs [17] and graphs of bounded degeneracy [3]. The CDS problem is also known to be W[2]-hard on general graphs [19], but admits a polynomial kernel on planar graphs, and more generally, on apex-minor-free graphs [22, 30, 41]. The problem is FPT on graphs of bounded degeneracy [29]. Cygan et al. [14] showed that CDS has no polynomial kernel even on 2-degenerate graphs unless NP $\subseteq$ co-NP/poly. Misra et al. [44] studied the effect of the girth of the input graph on the complexity of CDS, and showed that CDS remains W[1]-hard on graphs of girth 3 and 4, admits a fixed-parameter tractable algorithm but no polynomial kernel (unless NP $\subseteq$ co-NP/poly) on graphs of girth 5 and 6, and admits a polynomial kernel on graphs of girth at least 7. Fomin et al. [23] showed that both Dominating Set and CDS admit linear kernels on graphs with excluded topological minors. We refer the reader to [23] for a historical overview of the literature on these problems.

The Perfect Code problem, also called Efficient Domination or Perfect Domination, is known to be W[1]-complete [12, 18], and remains W[1]-hard even on bipartite graphs of girth 4 [34], but admits a polynomial kernel on planar graphs [31] and graphs of girth at least 5 [34]. Dawar and Kreutzer [15] showed that Perfect Code is fixed-parameter tractable on effectively nowhere dense graphs. For a summary of results on the (classical) complexity of Perfect Code on various graph classes, see [43].

The Partial Vertex Cover (PVC) problem, the "partial variant" of the widely-studied Vertex Cover problem, asks if t edges of a graph can be covered using k vertices. Both PVC and PDS have been studied w.r.t. the two natural parameters: k and t. When parameterized by k, unlike the widely-studied Vertex Cover, PVC is W[1]-hard on general graphs [32], and remains NP-hard even on bipartite graphs [5]. But Amini et al. [4], using a nuanced branching strategy called implicit branching, showed that PVC admits fixed-parameter tractable algorithms on graph classes with "large independent sets." In particular, they showed that PVC (parameterized by k) is FPT on bipartite graphs, triangle-free graphs, and H-minor free graphs, and thus, in particular, on planar graphs and graphs of bounded genus. As for PDS, note that a PDS instance with $t = n$ is precisely the Dominating Set problem, and therefore, the $W[2]$-hardness of Dominating Set (w.r.t. the parameter k) extends to PDS as well. And in contrast to Dominating Set, PDS remains W[1]-hard even on graphs of bounded degeneracy [29]. But the results due to Amini et al. [4] for a more general problem called Weighted Partial-(k, r, t)-Center showed that PDS, in particular, is FPT on planar graphs, graphs of bounded genus and graphs of bounded maximum degree. When parameterized by t, both PVC and PDS are FPTon general graphs [6, 11, 36, 37].

Related work on c-closed graphs. As mentioned earlier, Fox et al. [26] showed that every n-vertex c-closed graph contains at most $3^{c/3} \cdot n^2$ maximal cliques, and that all maximal cliques can be enumerated in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. In a preprint announced in 2020, Husic and Roughgarden [33] showed that instead of cliques, other "dense subgraphs" can be enumerated in time $f(c) \cdot \mathrm{poly}(n)$ as well. In particular, they showed that the problems of finding and enumerating subgraphs of bounded co-degree, bounded co-degeneracy and bounded co-treewidth in a c-closed graph admit algorithms that run in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. This result was soon followed by the work of Koana and Nichterlein [40], who investigated the complexity of enumerating all copies of a (small) fixed-graph H in a given c-closed graph. Note that for each fixed graph H, by brute-force, we can detect and enumerate all copies of H in a given n-vertex graph in time $n^{\mathcal{O}(|V(H)|)}$. Nonetheless, Koana and Nichterlein [40] designed significantly better combinatorial algorithms for such problems. They showed that for small graphs (i.e., graphs on 3 or 4 vertices) H, the H-detection and enumeration problems admit "FPT in P" algorithms [28] w.r.t. the parameter c, i.e., algorithms with runtime $\mathcal{O}(c^\ell n^i m^j)$ or $\mathcal{O}(c^\ell n^i + m^j)$, where m and n respectively are the number of edges and vertices of the input graph G, $c = cl(G)$, and ℓ, i and j are small constants independent of c and H. In particular, they designed such algorithms for 11 out of the 15 graphs on 3 or 4 vertices.

Related work on weakly γ-closed graphs. Along with c-closed graphs, Fox et al. [26] had also introduced a larger class of graphs called weakly γ-closed graphs. For a positive integer γ, a graph G is weakly γ-closed if every induced subgraph G' of G has a vertex v such that $|N_{G'}(v) \cap N_{G'}(u)| < \gamma$ for each $u \in V(G')$ with $u \neq v$ and $uv \notin E(G')$. Note that if a graph G is c-closed, then G is weakly c-closed as well. In a subsequent work, Koana et al. [38] extended their result for Independent Set in [39] to weakly γ-closed graphs. They showed that Independent Set admits a polynomial kernel on weakly γ-closed graphs as well. And they showed that a similar result holds for the $\mathcal{G}$-Subgraph problem, for a fixed family of graphs $\mathcal{G}$ that is closed under subgraphs, where the goal is to check if a given graph G contains an induced subgraph on at least k vertices that belongs to $\mathcal{G}$. Notice that Independent Set is a special case of $\mathcal{G}$-Subgraph with $\mathcal{G}$ being the family of all edgeless graphs. Koana et al. [38] also showed that two variants of Dominating Set, namely, Independent Dominating Set and Dominating Clique, are FPT on weakly γ-closed graphs. But they left open the complexity of Dominating Set on weakly γ-closed graphs, which was recently shown to be FPT by Lokshtanov and Surianarayanan [42]. Koana et al. [38] also gave bounds and enumeration algorithms for various choices of "dense subgraphs" in weakly γ-closed subgraphs. See [38, Table 1] for an overview of their results.

Due to space constraints, we only present our kernel for Perfect Code here. We omit other results and the proofs of statements marked with a ♣.

2 Preliminaries

For a positive integer ℓ, we denote the set $\{1, \ldots, \ell\}$ by $[\ell]$. We define the functions $\alpha, \beta : \mathbb{N} \to \mathbb{N}$ as follows: $\alpha(a, b) = (a-1)b+1$ and $\beta(a, b) = 2[(a-1)(b-1)+1]$ for every $a, b \in \mathbb{N}$. All graphs in this paper are simple and undirected. For a graph G, $V(G)$ and $E(G)$ respectively denote the vertex set and edge set of G. For a vertex $v \in V(G)$, $N_G(v)$ and $N_G[v]$ respectively denote the open and closed neighbourhood of v in G. Also, $d_G(v)$ denotes the degree of v in G, i.e., $d_G(v) = |N_G(v)|$. For a set $V' \subseteq V(G)$, $N_G(V')$ and $N_G[V']$ respectively denote the open neighbourhood and closed neighbourhood of V', i.e., $N_G(V') = (\bigcup_{v \in V'} N_G(v)) \setminus V'$ and $N_G[V'] = \bigcup_{v \in V'} N_G[v]$. And $CN_G(V')$ denotes the common neighbours of the vertices in V', i.e., $CN_G(V') = \bigcap_{v \in V'} N_G(v)$. Note that $CN_G(V') \subseteq V(G) \setminus V'$, because for every $v \in V'$, we have $v \notin N_G(v)$, and therefore, $v \notin CN_G(V')$. Also, for $V' \subseteq V(G)$ with $|V'| \geq 2$, by $N_G^{[2]}(V')$, we denote the union of the sets of common neighbours of every pair of vertices in V', i.e., $N_G^{[2]}(V') = (\bigcup_{\substack{u,v \in V' \\ u \neq v}} CN_G(\{u, v\})) \setminus V'$. For a pair of vertices $u, v \in V(G)$, $\text{dist}_G(x, y)$ denotes the length of a shortest path between x and y in G. We may omit the subscript when the graph G is clear from the context.

Consider a graph G. By a maximal clique (resp. maximal independent set) in G, we mean an inclusion-wise vertex maximal clique (resp. independent set) in G. That is, a clique Q (resp. an independent set I) in G is a maximal clique (resp. a maximal independent set) if $G[V(Q) \cup \{v\}]$ is not a clique (resp. $I \cup \{v\}$ is not an independent set) for any $v \in V(G) \setminus V(Q)$ (resp. $v \in V(G) \setminus I$). We say that an independent set I in G is 2-maximal if I is a maximal independent set and $(I \setminus \{v\}) \cup \{u, u'\}$ is not an independent set for every $v \in I$ and $u, u' \in V(G)$. That is, I is 2-maximal if I is maximal and no vertex in I can be replaced by 2 vertices from $V(G) \setminus I$.

We use $\mathcal{Q}(G)$ to denote the family of all maximal cliques in G. For $\ell > 0$, we denote by $\mathcal{Q}^\ell(G)$, the family of all maximal cliques in G of size at least ℓ. We also define two vertex subsets as follows: $L^\ell(G) = \bigcup_{Q \in \mathcal{Q}^\ell(G)} V(Q)$, and $R^\ell(G) = V(G) \setminus L^\ell(G)$. That is, $L^\ell(G)$ is the set of all vertices in G that belong to at least one maximal clique of size at least ℓ, and $R^\ell(G)$ contains the remaining vertices. Notice that $\left\{L^\ell(G), R^\ell(G)\right\}$ is a partition of $V(G)$ (with one of the parts possibly being empty).

2.1 Summary of Results From [26] and [39]

In this section, we briefly summarise the results from [26] and [39] that we will be using throughout. Following the notation of Koana et al. [39], for positive integers a, b and c, we let $R_c(a, b) = (c-1)\binom{b-1}{2} + (a-1)(b-1) + 1$.

▶ **Lemma 1** ([39]). *For positive integers a, b and c, every c-closed graph with at least $R_c(a, b)$ vertices contains either a clique of size a or an independent set of size b.*

▶ Remark 2. The proof of the above lemma [39, Proof of Lemma 3.1], in fact, shows that if G is a c-closed graph on at least $R_c(a, b)$ vertices such that G contains no clique of size a, then any 2-maximal independent set in G has size at least b.

Recall that the Independent Set problem takes a graph G and a non-negative integer k as input, and the task is to decide if G has an independent set of size at least k. Koana et al. [39] also showed that the Independent Set problem on c-closed graphs admits a kernel with ck^2 vertices. Specifically, they proved the following.

▶ **Lemma 3** ([39]). *There is an algorithm that, given a graph G and a non-negative integer k as input, runs in polynomial time, and outputs a graph G' such that (i) G' is an induced subgraph of G, (ii) G has an independent set of size k if and only if G' has an independent set of size k, and (iii) if $|V(G')| > ck^2$ then any maximal independent set in G' has size at least k.*

▶ **Corollary 4** (♣). *There is an algorithm that, given an n-vertex c-closed graph G and a non-negative integer k as input, runs in time $2^{\mathcal{O}(k \log(ck))} n^{\mathcal{O}(1)}$, and either returns a k-sized independent set of G if one exists, or correctly reports that no such set exists.*

Note that Corollary 4 follows immediately from Lemma 3. Fox et al. [26] showed that the number of maximal cliques in an n-vertex c-closed graph is bounded by $2^{\mathcal{O}(c)} n^2$. Specifically, they proved the following.

▶ **Lemma 5** ([26]). *Let G be a c-closed graph on n vertices. Then G contains at most $3^{(c-1)/3} n^2$ maximal cliques. Moreover, there is an algorithm that, given G as input, runs in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$, and enumerates all maximal cliques in G.*

2.2 Some Preliminary Lemmas

We now prove a few lemmas that we will be using throughout this paper.

▶ **Lemma 6** ([39]). *Let G be a c-closed graph, and Q a maximal clique in G. Then, for any $v \in V(G) \setminus V(Q)$, v has at most $c-1$ neighbours in $V(Q)$, i.e., $|N(v) \cap V(Q)| \leq c-1$.*

Lemma 6 implies that in a c-closed graph, every "small" dominating set must intersect every "large" clique.

▶ **Corollary 7** (♣). *Let G be a c-closed graph and k a non-negative integer. Let D be a dominating set of G of size at most k, and C a maximal clique in G of size at least $(c-1)k+1$. Then, $D \cap V(C) \neq \emptyset$.*

We now show that if a c-closed graph G contains sufficiently many large cliques, then G contains a sufficiently large independent set as well.

▶ **Lemma 8** (♣). *Let ℓ be a positive integer, and G be a c-closed graph such that $|\mathcal{Q}^{\beta(c,\ell)}(G)| \geq \ell$. Then, G has an independent set of size ℓ. Moreover, there is a polynomial time algorithm that, given a c-closed graph G and distinct $Q_1, Q_2, \ldots, Q_\ell \in \mathcal{Q}^{\beta(c,\ell)}(G)$ as input, returns an ℓ-sized independent set in G.*

▶ **Lemma 9** (♣). *Let ℓ be a positive integer. Let G be a graph and $V_1, V_2, \ldots, V_\ell \subseteq V(G)$ be such that $\bigcup_{i \in [\ell]} V_i = V(G)$, and $G[V_i]$ is a clique for every $i \in [\ell]$. Then, any independent set in G has size at most ℓ.*

▶ **Lemma 10.** *There is an algorithm that, given an n-vertex c-closed graph G and a positive integer ℓ as input, runs in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$, and either returns an independent set of size at least ℓ, or correctly concludes that every independent set in G has size at most $(\ell-1) + R_c(\beta(c,\ell),\ell) - 1 = \mathcal{O}(c \cdot \ell^2)$.*

Proof. Given G and ℓ as input, our algorithm works as follows. We first use the algorithm in Lemma 5 to construct $\mathcal{Q}(G)$ and $\mathcal{Q}^{\beta(c,\ell)}(G)$ in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. If $|\mathcal{Q}^{\beta(c,\ell)}(G)| \geq \ell$, then we return an ℓ-sized independent set constructed using the algorithm in Lemma 8.

Otherwise we construct the sets $L^{\beta(c,\ell)}(G)$, and $R^{\beta(c,\ell)}(G)$. By the definition of the sets $L^{\beta(c,\ell)}(G)$, and $R^{\beta(c,\ell)}(G)$, the induced subgraph $G' = G[R^{\beta(c,\ell)}(G)]$ contains no clique of size $\beta(c,\ell)$. And G', being an induced subgraph of G, is c-closed. So, if $|V(G')| \geq R_c(\beta(c,\ell),\ell)$, then by Lemma 1, G' contains an independent set of size ℓ. And we return a 2-maximal independent set in G', which can be computed in polynomial time, and which, by Remark 2, has size at least ℓ.

Otherwise, if $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell - 1$, and $|V(G')| = |R^{\beta(c,\ell)}(G)| \leq R_c(\beta(c,\ell),\ell) - 1$, then we return that every independent set in G has size at most $(\ell-1) + R_c(\beta(c,\ell),\ell) - 1$.

Note that the only time consuming step in this algorithm is the construction of the families $\mathcal{Q}(G)$ and $\mathcal{Q}^{\beta(c,\ell)}(G)$ in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. The rest of the steps run in polynomial time.

To see the correctness of the last step, assume that $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell - 1$ and $|V(G')| = |R^{\beta(c,\ell)}(G)| \leq R_c(\beta(c,\ell),\ell) - 1$. Note that by definition, $L^{\beta(c,\ell)}(G) = \bigcup_{Q \in \mathcal{Q}^{\beta(c,\ell)}(G)} V(Q)$. And therefore, by Lemma 9, any independent set in $G[L^{\beta(c,\ell)}(G)]$ has size at most $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell - 1$. Finally, as $\left\{L^{\beta(c,\ell)}(G), R^{\beta(c,\ell)}(G)\right\}$ is a partition of $V(G)$, for any independent set $I \subseteq V(G)$, we have $|I| = |I \cap L^{\beta(c,\ell)}(G)| + |I \cap R^{\beta(c,\ell)}(G)| \leq (\ell-1) + |R^{\beta(c,\ell)}(G)| \leq (\ell-1) + R_c(\beta(c,\ell),\ell) - 1$. Hence, the lemma follows. ◀

▶ **Lemma 11** (♣). *Let G be a graph and k a non-negative integer. Let I be an independent set in G of size $k+1$. Then, for any dominating set D of G, if $|D| \leq k$, then $D \cap N^{[2]}(I) \neq \emptyset$. Moreover, if G is c-closed, then $|N^{[2]}(I)| \leq (c-1)\binom{k+1}{2}$.*

▶ **Lemma 12** (♣). *Let G be a c-closed graph, and $Y \subseteq V(G)$ be such that $|Y| \leq c-1$. Then, the graph $G[CN(Y)]$ is $(c-|Y|)$-closed.*

3 A Polynomial Kernel for Perfect Code on c-closed graphs

To design our kernel, we consider a slightly more general version of the problem, which we call BW-Perfect Code. A bw-graph is a graph G along with a partition of $V(G)$ into two parts, B and W. We do not require that both B and W be non-empty. We call the elements of B black vertices and the elements of W white vertices, and for convenience we write that (G, B, W) is a bw-graph. A bw-perfect code of (G, B, W) is a set of vertices $D \subseteq B$ such that $|N[v] \cap D| = 1$ for every $v \in V(G)$. That is, a bw-perfect code is a set of black vertices that dominates every vertex of G exactly once. The definition of a perfect code immediately implies the following observation.

▶ **Observation 13.** *Let (G, B, W) be a bw-graph, and $D \subseteq B$ a bw-perfect code of G. Then,* (i) *D is a dominating set of G, and* (ii) *$dist_G(x, y) \geq 3$ for every pair of distinct vertices $x, y \in D$.*

We now formally define the BW-Perfect Code problem below.

BW-Perfect Code **Parameter:** $k + cl(G)$
Input: A bw-graph (G, B, W) and a non-negative integer k.
Question: Does (G, B, W) have a bw-perfect code of size at most k?

It is not difficult to see that an instance (G, k) of Perfect Code can be reduced to an equivalent instance $((G, B, W), k)$ of BW-Perfect Code by taking $B = V(G)$ and $W = \emptyset$. We now move to designing a kernel for BW-Perfect Code on c-closed graphs. We first prove that for each fixed positive integer c, the BW-Perfect Code problem on c-closed graphs admits a kernel with $\mathcal{O}(k^{3(2^c-1)})$ vertices. And then argue that an instance of BW-Perfect Code can be reduced in polynomial time to an equivalent instance of Perfect Code, which will give us the required kernel. Specifically, we prove the following theorem.

▶ **Theorem 14.** *Let c be a fixed positive integer. There is an algorithm that, when given an instance $((G, B, W), k)$ of* BW-Perfect Code *as input, where G is an n-vertex c-closed graph, runs in polynomial time, and returns an equivalent instance $((G', B', W'), k')$ of the* BW-Perfect Code *problem such that G' is a c-closed graph and $|V(G')| + k' = \mathcal{O}(k^{3(2^c-1)})$.*

In addition to Theorem 14, we also need the following two intermediate lemmas to prove that Perfect Code admits a kernel. The first of these lemmas deals with the Perfect Code problem on 1-closed graphs, (which are precisely graphs in which every connected component is a clique), and the second one presents a polynomial time reduction from BW-Perfect Code to Perfect Code.

▶ **Lemma 15** (♣). Perfect Code *is polynomial time solvable on 1-closed graphs.*

▶ **Lemma 16.** *Let $c > 1$ be a fixed integer. There is an algorithm that given an instance $((G', B', W'), k')$ of* BW-Perfect Code, *runs in polynomial time, and returns an equivalent instance (G'', k'') of* Perfect Code *such that (i) G'' is c-closed if G' is c-closed, (ii) $|V(G'')| = \mathcal{O}(|V(G')|)$, and (ii) $k'' \leq k' + 1$.*

Finally, as a consequence of Theorem 14, Lemmas 15 and 16, we derive the following.

▶ **Theorem 17.** *Let c be a fixed positive integer.* Perfect Code *on c-closed graphs admits a kernel with $\mathcal{O}(k^{3(2^c-1)})$ vertices.*

Proof. Let (G, k) be an instance of Perfect Code, where G is a c-closed graph. Our kernelization algorithm returns an equivalent instance (G'', k'') of Perfect Code as follows. If $c = 1$, then we use the algorithm in Lemma 15 to solve the Perfect Code problem on (G, k). And if (G, k) is a yes-instance, we take (G'', k'') to be a trivial yes-instance of Perfect Code with $|V(G'')| + k'' = \mathcal{O}(k)$, and otherwise we take (G'', k'') to be a trivial no-instance of Perfect Code with $|V(G'')| + k'' = \mathcal{O}(k)$, and return (G'', k'').

If $c > 1$, then we create from (G, k), an equivalent instance $((G, B, W), k)$ of BW-Perfect Code by taking $B = V(G)$ and $W = \emptyset$. And then apply the algorithm in Theorem 14, to obtain an equivalent instance $((G', B', W'), k')$ of BW-Perfect Code, where $|V(G')| + k' = \mathcal{O}(k^{3(2^c-1)})$. Finally, we apply the algorithm in Lemma 16 to obtain from $((G', B', W'), k')$ an equivalent instance (G'', k'') of Perfect Code. Note that as the algorithms in Lemma 15, Theorem 14 and Lemma 16, run in polynomial time, our kernelization algorithm returns (G'', k'') in polynomial time. And since Lemma 16 guarantees that $|V(G'')| = \mathcal{O}(|V(G')|)$, and $k'' \leq k' + 1$, we have $|V(G'')| + k'' = \mathcal{O}(k^{3(2^c-1)})$, and the theorem follows. ◀

So now we only need to prove Theorem 14. We first give a sketch of the proof of Lemma 16.

Proof Sketch of Lemma 16. Consider an instance $((G', B', W'), k')$ of BW-Perfect Code. If $W' = \emptyset$, then we take $G'' = G'$ and $k'' = k'$. Note that this choice of G'' and k'' satisfies all the properties stated in the lemma. So, assume that $W' \neq \emptyset$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$, and without loss of generality let $W' = \{v_1, v_2, \ldots, v_r\}$ for some $r \leq n$. We define the graph G'' as follows: $V(G'') = X \cup Y \cup Z$ and $E(G'') = E_1 \cup E_2 \cup E_3 \cup E_4$, where $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_r\}$ and $Z = \{z, z_1, z_2, \ldots, z_{k'+2}\}$; and $E_1 = \{x_i x_j \mid v_i v_j \in E(G')\}$, $E_2 = \{x_i y_i \mid i \in [r]\}$ and $E_3 = \{y_i z \mid i \in [r]\}$ and $E_4 = \{z z_i \mid i \in [k' + 2]\}$. And we set $k'' = k' + 1$. Note that $G''[X]$ is an isomorphic copy of G'. The set Y is another copy of W'. Thus, $\{x_1, x_2, \ldots, x_r\}$ and Y are two copies of W', and the set E_2 is a matching in G'' between the two copies.

First, $|V(G'')| = |X| + |Y| + |Z| = |V(G')| + |W'| + (k' + 3) = \mathcal{O}(|V(G')|)$. Second, we can show that $((G', B', W'), k')$ is a yes-instance of BW-Perfect Code if and only if (G'', k'') is a yes-instance of Perfect Code. ◀

The rest of this section is dedicated to proving Theorem 14. To that end, we first define two functions $\gamma, \mu : \mathbb{N} \to \mathbb{N}$ as follows. (Recall that $\alpha(a, b) = (a - 1)b + 1$ and $\beta(a, b) = 2[(a-1)(b-1)+1]$.) For $a, b \in \mathbb{N}$, we have $\gamma(1, b) = b+1$, and $\gamma(a, b) = b\mu(a-1, b)+1$; and $\mu(a, b) = \gamma(a, b) + R_a(\beta(a, \gamma(a, b) + 1), \gamma(a, b) + 1) - 1$. These functions γ and μ will be used to bound the size of independent sets in G when $((G, B, W), k)$ is a yes-instance.

▶ **Observation 18.** *Observe that for every fixed $a, i \in \mathbb{N}$, and for $b \in \mathbb{N}$, we have $R_i(a, b) = \mathcal{O}(b^2)$ and $\beta(a, b) = \mathcal{O}(b)$. Therefore, we have*

$$\begin{array}{ll}
\gamma(1, b) = \mathcal{O}(b) & \mu(1, b) = \mathcal{O}(b) + R_1(\mathcal{O}(b), \mathcal{O}(b)) = \mathcal{O}(b^2) \\
\gamma(2, b) = b\mu(1, b) + 1 = \mathcal{O}(b^3) & \mu(2, b) = \mathcal{O}(b^3) + R_2(\mathcal{O}(b^3), \mathcal{O}(b^3)) = \mathcal{O}(b^6) \\
\gamma(3, b) = b\mu(2, b) + 1 = \mathcal{O}(b^7) & \mu(3, b) = \mathcal{O}(b^7) + R_3(\mathcal{O}(b^7), \mathcal{O}(b^7)) = \mathcal{O}(b^{14}) \\
\cdots & \cdots \\
\gamma(a, b) = \mathcal{O}(b^{2^a-1}) & \mu(a, b) = \mathcal{O}(b^{2(2^a-1)}).
\end{array}$$

Outline of the kernel. Our kernel for BW-Perfect Code has two parts. In the first part, we bound the size of independent sets in (G, B, W) using Reduction Rule 19, and in the second part, we bound the size of cliques in (G, B, W) using Reduction Rules 27-29. Once the size of cliques and independent sets are bounded, we apply Lemma 1.

To bound the size of independent sets in case $((G, B, W), k)$ is a yes-instance, observe the following fact. Consider an independent set I in G and a bw-perfect code $D \subseteq B$ of size at most k. Then, we can partition I into at most k parts, say, $I_1, I_2, \ldots, I_k$, such that for each $j \in [k]$, there exists a unique vertex $v_j \in D$ that dominates I_j, i.e., $I_j \subseteq N(v_j)$. Thus, to bound $|I|$, we only need to bound $|I_j|$ for every $j \in [k]$. More generally, we only need to bound the size of independent sets contained in $N(v)$ for every $v \in V(G)$. To do this, suppose that for every $Y \subseteq V(G)$ with $|Y| = 2$ we have already managed to bound the size of independent sets contained in $CN(Y)$ by some function of c and k, say, $f(c, k)$. That is, every independent set with at least 2 common neighbours has size at most $f(c, k)$. Now, consider $v \in V(G)$. And let I' be an independent set of size at least $k \cdot f(c, k) + 1$ contained in $N(v)$ and D a bw-perfect code of size at most k. Then, we must have $v \in D$. If not, there exists $u \in D$ that dominates at least $|I'|/k$ vertices of I. That is, there exist $u \in D$ and $I'' \subseteq I'$ such that $|I''| \geq |I'|/k > f(c, k)$ and $I'' \subseteq N(u)$. But note that $I'' \subseteq I' \subseteq N(v)$. Thus, $I'' \subseteq CN(\{u, v\})$ and $|I''| > f(c, k)$, which we have already ruled out to be impossible. By repeating these arguments, we can show that, to obtain the bound of $f(c, k)$ for independent sets with 2 common neighbours, we only need to bound the size of independent sets with 3 common neighbours. This train of arguments only needs to continue until we reach independent sets with $c - 1$ common neighbours. Thus, we start with sets Y of size $c - 1$ and bound the size of independent sets contained in $CN(Y)$. Then proceed to sets Y of size $c - 2$ and so on. This idea is formalised in Reduction Rule 19. But the difficulty comes in checking if $CN(Y)$ contains an independent set of the required size, which cannot be done in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. To overcome this, we use the weaker result of Lemma 10, which causes the bound on the independent set size to increase exponentially in each successive stage. Thus, after $c - 1$ stages, we only manage to obtain a bound of $\mu(c - 1, k) = k^{\mathcal{O}(2^c)}$ for the size of independent sets contained in $N(v)$ for every $v \in V(G)$. And this bound is where the kernel size comes from.

In the second part, bounding the clique size is fairly straightforward. This involves removing twin vertices (Reduction Rule 27), and identifying irrelevant vertices (vertices that cannot belong to any bw-perfect code of size at most k) and colouring them white or removing them (Reduction Rules 28 and 29). (Also, each time we introduce a reduction rule, we apply it exhaustively. So from that point onwards, we would assume that the reduction rule is no longer applicable.)

We now formally introduce the following reduction rule.

▶ **Reduction Rule 19.** *For each $i \in [c - 1]$, we introduce Reduction Rule 19.i as follows. Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *For each fixed set $Y \subseteq V(G)$ with $|Y| = c - i$, we run the algorithm in Lemma 10 on the graph $G[CN(Y)]$ with $\ell = \gamma(i, k) + 1$. If the algorithm returns an independent set I of size ℓ, then delete a vertex $v \in I$ from G, and colour $N_G(v) \setminus Y$ white. That is, we create a new instance $((G', B', W'), k)$ as follows: $G' = G - v$, $B' = B \setminus (N_G[v] \setminus Y)$ and $W' = V(G') \setminus B' = W \cup (N_G[v] \setminus Y)$. We keep repeating this proceudre until the algorithm in Lemma 10 returns that every independent set in $G[CN(Y)]$ has size at most $(\ell - 1) + R_c(\beta(c, \ell), \ell) - 1$. Also, we apply Reduction Rule 19.i in the increasing order of i. That is, we first apply Reduction Rule 19.*1 *exhaustively, and for each $i \in [c - 1] \setminus \{1\}$, we apply Reduction Rule 19.i only if Reduction Rule 19.$(i - 1)$ is no longer applicable.*

We now observe the following fact, which will be useful in establishing the correctness of Reduction Rule 19.

▶ **Observation 20.** *Fix $i \in [c-1]$. For any $Y \subseteq V(G)$ with $|Y| = c - i$, by Lemma 12, the subgraph $G[CN(Y)]$ is i-closed. Therefore, after an exhaustive application of Reduction Rule 19.i, by Lemma 10, every independent set in $G[CN(Y)]$ has size at most $\gamma(i,k) + R_i(\beta(i,\gamma(i,k)+1), \gamma(i,k)+1) - 1 = \mu(i,k)$. In particular, when $i = c-1$, we get that after an exhaustive application of Reduction Rule 19.$(c-1)$, for every $v \in V(G)$, every independent set in $G[N(v)]$ has size at most $\mu(c-1,k)$.*

▶ **Lemma 21.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code. *Let $Y \subseteq V(G)$ be such that $|Y| = c-1$, and $I \subseteq CN(Y)$ be an independent set with $|I| \geq \gamma(1,k)$. Then, for any bw-perfect code $D \subseteq B$ of (G,B,W) with $|D| \leq k$, we have $|D \cap Y| = 1$.*

Proof. Let $D \subseteq B$ be a bw-perfect code of (G,B,W) with $|D| \leq k$. We first claim that $D \cap Y \neq \emptyset$. Assume for a contradiction that $D \cap Y = \emptyset$. Now, since $|I| \geq \gamma(1,k) = k+1$ and $|D| \leq k$, by the pigeonhole principle, there exists a vertex $u \in D$ that dominates at least two vertices of I, say, $w_1, w_2 \in I$. That is, $u \in N[w_1] \cap N[w_2]$. Since I is an independent set, and $uw_1, uw_2 \in E(G)$, we can conclude that $u \neq w_1$ and $u \neq w_2$. Thus, $u \in N(w_1) \cap N(w_2)$. But since since $w_1, w_2 \in I \subseteq CN(Y)$, we get that $Y \subseteq N(w_1) \cap N(w_2)$. Thus, $Y \cup \{u\} \subseteq N(w_1) \cap N(w_2)$. Because of our assumption that $D \cap Y = \emptyset$, we have $u \notin Y$, and thus $|Y \cup \{u\}| = c$. Thus, w_1 and w_2 have at least c common neighbours, and therefore $w_1w_2 \in E(G)$, which is not possible as $w_1$and w_2 belong to the independent set I. Thus, $D \cap Y \neq \emptyset$. Now, if there exist $y_1, y_2 \in D \cap Y$, where $y_1 \neq y_2$, then for any $x \in I$, we have $y_1, y_2 \in N[x] \cap D$, which, by the definition of a bw-perfect code, is not possible. Therefore, we conclude that $|D \cap Y| = 1$. ◀

▶ **Lemma 22.** *Fix $i \in [c-1] \setminus \{1\}$. Let $((G,B,W),k)$ be an instance of* BW-Perfect Code *to which Reduction Rule 19.$(i-1)$ has been applied exhaustively. Let $Y \subseteq V(G)$ be such that $|Y| = c-i$, and $I \subseteq CN(Y)$ be an independent set with $|I| \geq \gamma(i,k)$. Then, for any bw-perfect code $D \subseteq B$ of (G,B,W) with $|D| \leq k$, we have $|D \cap Y| = 1$.*

Proof. Let $D \subseteq B$ be a bw-perfect code of (G,B,W) with $|D| \leq k$. We first claim that $D \cap Y \neq \emptyset$. Assume for a contradiction that $D \cap Y = \emptyset$. Now, since $|I| \geq \gamma(i,k) = k\mu(i-1,k)+1$ and $|D| \leq k$, by the pigeonhole principle, there exists a vertex $u \in D$ that dominates at least $\mu(i-1,k)+1$ vertices of I. Let $I' \subseteq I$ be such that $|I'| \geq \mu(i-1,k)+1$ and u dominates I'. That is, $I' \subseteq N[u]$. Observe first that $u \notin I'$. To see this, suppose that $u \in I'$. Then, for every $w \in I' \setminus \{u\}$, since u dominates w, we must have $uw \in E(G)$, which contradicts the fact that I' is an independent set. So, $u \notin I'$, and therefore, $I' \subseteq N(u)$. And we already have $I' \subseteq I \subseteq CN(Y)$. We can conclude that $I' \subseteq N(u) \cap CN(Y) = CN(Y \cup \{u\})$. Because of our assumption that $D \cap Y = \emptyset$, we have $u \notin Y$, and thus $|Y \cup \{u\}| = c-i+1 = c-(i-1)$. That is, $Y \cup \{u\}$ is a set of size $c-(i-1)$, and I' is an independent set such that $I' \subseteq CN(Y \cup \{u\})$, and $|I'| \geq \mu(i-1,k)+1$. But this conclusion contradicts Observation 20 because of our assumption that Reduction Rule 19.$(i-1)$ has been applied exhaustively. Thus, $D \cap Y \neq \emptyset$. Now, if there exist $y_1, y_2 \in D \cap Y$, where $y_1 \neq y_2$, then for any $x \in I$, we have $y_1, y_2 \in N[x] \cap D$, which, by the definition of a bw-perfect code, is not possible. Therefore, we conclude that $|D \cap Y| = 1$. ◀

▶ **Lemma 23.** *Reduction Rule 19.i is safe.*

Proof. Let $((G',B',W'),k)$ be the instance obtained from $((G,B,W),k)$ by a single application of Reduction Rule 19.i. Then, there exists $Y \subseteq V(G)$ with $|Y| = c - i$, and an independent set $I \subseteq CN(Y)$ with $|I| = \gamma(i,k) + 1$ and a vertex $v \in I$ such that $G' = G - v$, $B' = B \setminus (N_G[v] \setminus Y)$ and $W' = V(G') \setminus B' = W \cup (N_G[v] \setminus Y)$. We shall show that $((G,B,W),k)$ and $((G',B',W'),k)$ are equivalent instances.

First consider the case when $i = 1$. Then, $|Y| = c - 1$, and $|I| = \gamma(1,k) + 1$. Assume that $((G,B,W),k)$ is a yes-instance of BW-Perfect Code, and let $D \subseteq B$ be a bw-perfect code of (G,B,W) of size at most k. Then, by Lemma 21, $|D \cap Y| = 1$. Let $\{y\} = D \cap Y$. But then since $y \in D$, and $I \subseteq CN(Y) \subseteq N(y)$, we have $I \cap D = \emptyset$. In particular $v \notin D$. Also, for any $w \in N_G(v) \setminus Y$, we have $\text{dist}_G(y,w) \leq 2$, and thus, by Observation 13, we have $w \notin D$. Thus, $D \cap (N_G[v] \setminus Y) = \emptyset$, and therefore, $D \subseteq B \setminus (N_G[v] \setminus Y) = B'$. Thus, D is a bw-perfect code of (G',B',W') as well.

Conversely, assume that $((G',B',W'),k)$ is a yes-instance, and let $D' \subseteq B'$ be a bw-perfect code of (G',B',W') with $|D'| \leq k$. We claim that D' is a bw-perfect code of (G,B,W) as well. Note that for any $x \in V(G) \setminus \{v\}$, we have $N_{G'}[x] = N_G[x] \setminus \{v\}$. Therefore, since $v \notin D'$, we have $|D' \cap N_G[x]| = |D' \cap N_{G'}[x]| = 1$. So, now we only need to show that $|D' \cap N_G[v]| = 1$. Note that $N_G[v] = (N_G[v] \setminus Y) \cup (N_G[v] \cap Y)$. First, since $N_G[v] \setminus Y \subseteq W'$, and $D' \subseteq B'$, we get that $D' \cap (N_G[v] \setminus Y) = \emptyset$. So we only need to show that $|D' \cap (N_G[v] \cap Y)| = 1$. Now, observe that as $|I \setminus \{v\}| = \gamma(1,k)$, by Lemma 21, we have $|D' \cap Y| = 1$. Let $\{y'\} = D' \cap Y$. Then, $y' \in D' \cap N_G[v]$, and in fact, $\{y'\} = D' \cap (N_G[v] \cap Y)$. This completes the proof for the case when $i = 1$.

Now, assume that $i > 1$. First, by assumption, Reduction Rule 19.j is not applicable to $((G,B,W),k)$ for any $j \in [i-1]$. And we have $|Y| = c-i$, and $|I| = \gamma(i-1,k)+1$. Assume that $((G,B,W),k)$ is a yes-instance of BW-Perfect Code, and let $D \subseteq B$ be a bw-perfect code of (G,B,W) of size at most k. Then, by Lemma 22, we have $|D \cap Y| = 1$. Let $\{y\} = D \cap Y$. But then since $y \in D$, and $I \subseteq CN(Y) \subseteq N(y)$, we have $I \cap D = \emptyset$. In particular $v \notin D$. Also, for any $w \in N_G(v) \setminus Y$, we have $\text{dist}_G(y,w) \leq 2$, and thus, by Observation 13, we have $w \notin D$. Thus, $D \cap (N_G[v] \setminus Y) = \emptyset$, and therefore, $D \subseteq B \setminus (N_G[v] \setminus Y) = B'$. Thus, D is a bw-perfect code of (G',B',W') as well.

Conversely, assume that $((G',B',W'),k)$ is a yes-instance, and let $D' \subseteq B'$ be a bw-perfect code of (G',B',W') with $|D'| \leq k$. We claim that D' is a bw-perfect code of (G,B,W) as well. Note that for any $x \in V(G) \setminus \{v\}$, we have $N_{G'}[x] = N_G[x] \setminus \{v\}$. Therefore, since $v \notin D'$, we have $|D' \cap N_G[x]| = |D' \cap N_{G'}[x]| = 1$. So, now we only need to show that $|D' \cap N_G[v]| = 1$. Note that $N_G[v] = (N_G[v] \setminus Y) \cup (N_G[v] \cap Y)$. First, since $N_G[v] \setminus Y \subseteq W'$, and $D' \subseteq B'$, we get that $D' \cap (N_G[v] \setminus Y) = \emptyset$. So we only need to show that $|D' \cap (N_G[v] \cap Y)| = 1$. Now, observe that as $|I \setminus \{v\}| = \gamma(i,k)$, by Lemma 22, we have $|D' \cap Y| = 1$. Let $\{y'\} = D' \cap Y$. Then, $y' \in D' \cap N_G[v]$, and in fact, $\{y'\} = D' \cap (N_G[v] \cap Y)$. This completes the proof for the lemma. ◀

▶ Remark 24. Observe that each application of Reduction Rule 19 can be executed in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. Also, for each set $Y \subseteq V(G)$ with $|Y| \leq c - 1$, Reduction Rule 19 is applied only at most $|CN(Y)| \leq n$ times. And note that the set Y has at most $\sum_{i=1}^{c-1} \binom{n}{i} = n^{\mathcal{O}(c)}$ choices. Thus, Reduction Rule 19 can be applied exhaustively in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(c)}$. Since c is a fixed constant, we have $2^{\mathcal{O}(c)} n^{\mathcal{O}(c)} = n^{\mathcal{O}(1)}$. That is, we can exhaustively apply Reduction Rule 19 in polynomial time. So, from now on, we assume that Reduction Rule 19 has been applied exhaustively.

The following lemma bounds the size of an independent set in G if $((G,B,W),k)$ is a yes-instance.

▶ **Lemma 25.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then every independent set in G has size at most $\gamma(c, k) - 1$.*

Proof. Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code, and let $D \subseteq B$ be a bw-perfect code of (G, B, W) of size at most k. Let $I \subseteq V(G)$ be an independent set. Assume for a contradiction that $|I| \geq \gamma(c, k) = k\mu(c-1, k) + 1$. Then, since $|D| \leq k$, by the pigeonhole principle, there exists $v \in D$ such that v dominates at least $\mu(c-1, k) + 1$ vertices of I. That is, there exists an independent set I' such that $I' \subseteq N(v)$ and $|I'| \geq \mu(c-1, k)+1$, which, by Observation 20, is not possible, as Reduction Rule 19, and in particular, Reduction Rule 19.$(c-1)$ has been applied exhaustively. ◀

We have thus bounded the size of every independent set in G for yes-instances. This immediately bounds the number of large cliques (by Lemma 8), as well as the number of vertices that do not belong to any large maximal clique (by Lemma 1).

▶ **Lemma 26.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then*
1. *$|\mathcal{Q}^{\beta(c,\gamma(c,k))}(G)| \leq \gamma(c, k) - 1$, and*
2. *$|R^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1$.*

Proof. Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code.
1. If $|\mathcal{Q}^{\beta(c,\gamma(c,k))}(G)| \geq \gamma(c, k)$, then by Lemma 8, G contains an independent set of size $\gamma(c, k)$, which contradicts Lemma 25.
2. By the definition of $R^{\beta(c,\gamma(c,k))}(G)$, the induced subgraph $G[R^{\beta(c,\gamma(c,k))}(G)]$ of G contains no clique of size $\beta(c, \gamma(c, k))$. By Lemma 25, the graph G, and hence the graph $G[R^{\beta(c,\gamma(c,k))}(G)]$, contains no independent set of size $\gamma(c, k)$. The bound then follows from Lemma 1. ◀

In the next three reduction rules we bound the size of every clique in G as well, which, in turn, will help us bound $|L^{\beta(c,\gamma(c,k))}(G)|$. We begin by introducing a reduction rule, which says that if two vertices have same closed neighborhood and the same colour, then we can safely delete one of them.

▶ **Reduction Rule 27.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *Let $x, y \in V(G)$ be distinct vertices such that $N_G[x] = N_G[y]$. If $x, y \in B$ or $x, y \in W$, then delete x.*

Let Q be a maximal clique of size at least $\alpha(c, k)$. By Corollary 7, exactly one vertex from $V(Q)$ is in every bw-perfect code. Therefore, no vertex from $N(V(Q))$ belongs to a bw-perfect code of size at most k. So we color $N(V(Q))$ white in the next reduction rule.

▶ **Reduction Rule 28.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code, *and let $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Colour $N(V(Q))$ white. That is, we construct the instance $((G, B', W'), k)$ of* BW-Perfect Code, *where $W' = W \cup N(V(Q))$, and $B' = B \setminus N(V(Q))$.*

Let $Q \in \mathcal{Q}^{\alpha(c,k)+1}(G)$. We define $Z(Q)$ to be the set of vertices in $V(Q)$ that have neighbours in some other maximal clique of size at least $\alpha(c, k)$, i.e., $Z(Q) = \{u \in V(Q) \mid uv \in E(G)$ for some $v \in V(Q')$, where $Q' \in \mathcal{Q}^{\alpha(c,k)}(G), u \notin V(Q')$, and $Q' \neq Q\}$. In the following reduction rule we show that we can safely delete $Z(Q)$.

▶ **Reduction Rule 29.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If there exists $Q \in \mathcal{Q}^{\alpha(c,k)+1}(G)$ and $v \in Z(Q)$, then delete v. That is, we construct the instance $((G', B', W'), k)$ of* BW-Perfect Code, *where $G' = G - v$, $B' = B \setminus \{v\}$, and $W' = W \setminus \{v\}$.*

▶ **Lemma 30 (♣).** *Reduction Rules 27, 28, and 29 are safe.*

▶ Remark 31. Observe that given an instance $((G, B, W), k)$ of BW-Perfect Code, using the algorithm in Lemma 5, we can construct $\mathcal{Q}^{\alpha(c,k)}(G)$ (and $\mathcal{Q}^{\alpha(c,k+1)}(G)$) in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. And once we construct these families of cliques, we can then exhaustively apply Reduction Rules 28 in time $|\mathcal{Q}^{\alpha(c,k)}(G)|n^{\mathcal{O}(1)}$ and Reduction Rule 29 in time $|\mathcal{Q}^{\alpha(c,k+1)}(G)|n^{\mathcal{O}(1)}$. Also, observe that we can exhaustively apply Reduction Rule 27 in polynomial time. So from now on, we assume that we have exhaustively applied Reduction Rules 27-29.

▶ **Lemma 32 (♣).** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then for every $Q \in \mathcal{Q}^{\beta(c,\gamma(c,k))}(G)$, we have*

1. *$Z(Q) = \emptyset$, and*
2. *$|V(Q)| \leq (c-1)[R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1] + 2$.*

Finally, Lemmas 26-(1) and 32-(2) together bound $|L^{\beta(c,\gamma(c,k))}(G)|$, which bounds $|V(G)|$.

▶ **Lemma 33.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then $|V(G)| = \mathcal{O}(k^{3(2^c-1)})$.*

Proof. Assume that $((G, B, W), k)$ is a yes-instance. Then, by Lemma 26-(1), we have $|\mathcal{Q}^{\beta(c,\gamma(c,k))}(G)| \leq \gamma(c, k) - 1 = \mathcal{O}(k^{2^c-1})$, and by Lemma 32-(2), we have $|V(Q)| \leq (c - 1)[R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1] + 2 = \mathcal{O}((\gamma(c, k))^2) = \mathcal{O}(k^{2(2^c-1)})$. Therefore, we have

$$\begin{aligned} |L^{\beta(c,\gamma(c,k))}(G)| &= |\bigcup_{Q \in \mathcal{Q}^{\beta(c,\gamma(c,k))}(G)} V(Q)| \\ &\leq (\gamma(c, k) - 1) \cdot (c - 1)[R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1] + 2 \\ &= \mathcal{O}(k^{2^c-1}) \cdot \mathcal{O}(k^{2(2^c-1)}) \\ &= \mathcal{O}(k^{3(2^c-1)}). \end{aligned}$$

Also, by Lemma 26-(2), we have $|R^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1 = R_c(\mathcal{O}(k^{2^c-1}), \mathcal{O}(k^{2^c-1})) = \mathcal{O}(k^{2(2^c-1)})$. Finally, since $\left\{L^{\beta(c,\gamma(c,k))}(G), R^{\beta(c,\gamma(c,k))}(G)\right\}$ is a partition of $V(G)$, we conclude that $|V(G)| = \mathcal{O}(k^{3(2^c-1)})$. ◀

Each of our reduction rules is safe and by Remarks 24 and 31, all the reduction rules we introduced can be executed in polynomial time, and are applied only polynomially many times. We have thus proved Theorem 14.

4 Conclusion

We resolved the parameterized complexity of three domination problems – Perfect Code, CDS and PDS– on c-closed graphs. We believe that our results, along with that of Koana et al. [39], make a convincing case for pursuing the closure of a graph as a significant structural parameter. We also believe that the arguments in this paper can be adapted to solve similar problems on c-closed graphs. In particular, our strategy for Perfect Code may be applicable to the Even Dominating Set (resp. Odd Dominating Set) problems, where the goal is to check if a graph G has a dominating set D of size at most k such that D dominates every vertex of G an even (resp. odd) number of times. While we showed that PDS is W[1]-hard even on 2-closed graphs, the status of Partial Vertex Cover on c-closed graphs still remains open. It would be interesting to see if any our results extend to weakly γ-closed graphs (see [26] and [38]) as well.

References

1 Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. doi:10.1007/s00453-001-0116-5.

2 Jochen Alber, Hongbing Fan, Michael R. Fellows, Henning Fernau, Rolf Niedermeier, Frances A. Rosamond, and Ulrike Stege. A refined search tree technique for dominating set on planar graphs. *J. Comput. Syst. Sci.*, 71(4):385–405, 2005. doi:10.1016/j.jcss.2004.03.007.

3 Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54(4):544–556, 2009. doi:10.1007/s00453-008-9204-0.

4 Omid Amini, Fedor V. Fomin, and Saket Saurabh. Implicit branching and parameterized partial cover problems. *J. Comput. Syst. Sci.*, 77(6):1159–1171, 2011. doi:10.1016/j.jcss.2010.12.002.

5 Nicola Apollonio and Bruno Simeone. The maximum vertex coverage problem on bipartite graphs. *Discret. Appl. Math.*, 165:37–48, 2014. doi:10.1016/j.dam.2013.05.015.

6 Markus Bläser. Computing small partial coverings. *Inf. Process. Lett.*, 85(6):327–331, 2003. doi:10.1016/S0020-0190(02)00434-9.

7 J. Adrian Bondy and Vasek Chvátal. A method in graph theory. *Discret. Math.*, 15(2):111–135, 1976. doi:10.1016/0012-365X(76)90078-9.

8 J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008. doi:10.1007/978-1-84628-970-5.

9 Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.

10 Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008. doi:10.1093/comjnl/bxm086.

11 Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2006. doi:10.1007/11847250_22.

12 Marco Cesati. Perfect code is W[1]-complete. *Inf. Process. Lett.*, 81(3):163–168, 2002. doi:10.1016/S0020-0190(01)00207-1.

13 Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2, 2006. doi:10.1145/1132952.1132954.

14 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Kernelization hardness of connectivity problems in d-degenerate graphs. *Discret. Appl. Math.*, 160(15):2131–2141, 2012. doi:10.1016/j.dam.2012.05.016.

15 Anuj Dawar and Stephan Kreutzer. Domination problems in nowhere-dense classes. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *LIPIcs*, pages 157–168. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2315.

16 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Trans. Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

17 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005. doi:10.1145/1101821.1101823.

18 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995. doi:10.1016/0304-3975(94)00097-3.

19 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

20 John A. Ellis, Hongbing Fan, and Michael R. Fellows. The dominating set problem is fixed parameter tractable for graphs of bounded genus. *J. Algorithms*, 52(2):152–168, 2004. doi:10.1016/j.jalgor.2004.02.001.

21 Michael R. Fellows and Mark N. Hoover. Perfect domination. *Australas. J Comb.*, 3:141–150, 1991. URL: http://ajc.maths.uq.edu.au/pdf/3/ocr-ajc-v3-p141.pdf.

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 503–510. SIAM, 2010. doi:10.1137/1.9781611973075.43.

23 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Kernels for (connected) dominating set on graphs with excluded topological minors. *ACM Trans. Algorithms*, 14(1):6:1–6:31, 2018. doi:10.1145/3155298.

24 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:10.1137/S0097539702419649.

25 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.55.

26 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM J. Comput.*, 49(2):448–464, 2020. doi:10.1137/18M1210459.

27 Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H. Freeman, New York, 1979.

28 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theor. Comput. Sci.*, 689:67–95, 2017. doi:10.1016/j.tcs.2017.05.017.

29 Petr A. Golovach and Yngve Villanger. Parameterized complexity for domination problems on degenerate graphs. In Hajo Broersma, Thomas Erlebach, Tom Friedetzky, and Daniël Paulusma, editors, *Graph-Theoretic Concepts in Computer Science, 34th International Workshop, WG 2008, Durham, UK, June 30 - July 2, 2008. Revised Papers*, volume 5344 of *Lecture Notes in Computer Science*, pages 195–205, 2008. doi:10.1007/978-3-540-92248-3_18.

30 Qianping Gu and Navid Imani. Connectivity is not a limit for kernelization: Planar connected dominating set. In Alejandro López-Ortiz, editor, *LATIN 2010: Theoretical Informatics, 9th Latin American Symposium, Oaxaca, Mexico, April 19-23, 2010. Proceedings*, volume 6034 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2010. doi:10.1007/978-3-642-12200-2_4.

31 Jiong Guo and Rolf Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 375–386. Springer, 2007. doi:10.1007/978-3-540-73420-8_34.

32 Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of vertex cover variants. *Theory Comput. Syst.*, 41(3):501–520, 2007. doi:10.1007/s00224-007-1309-3.

33 Edin Husic and Tim Roughgarden. FPT algorithms for finding dense subgraphs in c-closed graphs. *CoRR*, abs/2007.09768, 2020. arXiv:2007.09768.

34 Minghui Jiang and Yong Zhang. Perfect domination and small cycles. *Discret. Math. Algorithms Appl.*, 9(3):1750030:1–1750030:11, 2017. doi:10.1142/S1793830917500306.

35 Iyad A. Kanj and Ljubomir Perkovic. Improved parameterized algorithms for planar dominating set. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 399–410. Springer, 2002. doi:10.1007/3-540-45687-2_33.

36 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Intuitive algorithms and t-vertex cover. In Tetsuo Asano, editor, *Algorithms and Computation, 17th International Symposium, ISAAC 2006, Kolkata, India, December 18-20, 2006, Proceedings*, volume 4288 of *Lecture Notes in Computer Science*, pages 598–607. Springer, 2006. doi:10.1007/11940128_60.

37 Joachim Kneis, Daniel Mölle, and Peter Rossmanith. Partial vs. complete domination: t-dominating set. In Jan van Leeuwen, Giuseppe F. Italiano, Wiebe van der Hoek, Christoph Meinel, Harald Sack, and Frantisek Plasil, editors, *SOFSEM 2007: Theory and Practice of Computer Science, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings*, volume 4362 of *Lecture Notes in Computer Science*, pages 367–376. Springer, 2007. doi:10.1007/978-3-540-69507-3_31.

38 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Computing dense and sparse subgraphs of weakly closed graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.20.

39 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Exploiting c-closure in kernelization algorithms for graph problems. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 65:1–65:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.65.

40 Tomohiro Koana and André Nichterlein. Detecting and enumerating small induced subgraphs in c-closed graphs. *CoRR*, abs/2007.12077, 2020. arXiv:2007.12077.

41 Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. Linear kernel for planar connected dominating set. In Jianer Chen and S. Barry Cooper, editors, *Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009, Changsha, China, May 18-22, 2009. Proceedings*, volume 5532 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 2009. doi:10.1007/978-3-642-02017-9_31.

42 Daniel Lokshtanov and Vaishali Surianarayanan. Dominating set in weakly closed graphs is fixed parameter tractable. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.29.

43 Chin Lung Lu and Chuan Yi Tang. Weighted efficient domination problem on some perfect graphs. *Discret. Appl. Math.*, 117(1-3):163–182, 2002. doi:10.1016/S0166-218X(01)00184-6.

44 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. The kernelization complexity of connected domination in graphs with (no) small cycles. *Algorithmica*, 68(2):504–530, 2014. doi:10.1007/s00453-012-9681-z.

45 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. FPT algorithms for connected feedback vertex set. *J. Comb. Optim.*, 24(2):131–146, 2012. doi:10.1007/s10878-011-9394-2.

46 Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory Comput. Syst.*, 43(2):234–253, 2008. doi:10.1007/s00224-007-9089-3.

47 John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

48 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Trans. Algorithms*, 9(1):11:1–11:23, 2012. doi:10.1145/2390176.2390187.

49 Venkatesh Raman and Saket Saurabh. Short cycles make W -hard problems hard: FPT algorithms for W -hard problems in graphs with no short cycles. *Algorithmica*, 52(2):203–225, 2008. doi:10.1007/s00453-007-9148-9.

Obstructions for Matroids of Path-Width at most k and Graphs of Linear Rank-Width at most k

Mamadou Moustapha Kanté
Université Clermont Auvergne, Clermont Auvergne INP, LIMOS, CNRS, Aubière, France

Eun Jung Kim
Université Paris-Dauphine, PSL University, CNRS, UMR 7243, LAMSADE, Paris, France

O-joung Kwon
Department of Mathematics, Incheon National University, Incheon, South Korea
Discrete Mathematics Group, Institute for Basic Science (IBS), Daejeon, South Korea

Sang-il Oum
Discrete Mathematics Group, Institute for Basic Science (IBS), Daejeon, South Korea
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Abstract

Every minor-closed class of matroids of bounded branch-width can be characterized by a minimal list of excluded minors, but unlike graphs, this list could be infinite in general. However, for each fixed finite field $\mathbb{F}$, the list contains only finitely many $\mathbb{F}$-representable matroids, due to the well-quasi-ordering of $\mathbb{F}$-representable matroids of bounded branch-width under taking matroid minors [J. F. Geelen, A. M. H. Gerards, and G. Whittle (2002)]. But this proof is non-constructive and does not provide any algorithm for computing these $\mathbb{F}$-representable excluded minors in general.

We consider the class of matroids of path-width at most k for fixed k. We prove that for a finite field $\mathbb{F}$, every $\mathbb{F}$-representable excluded minor for the class of matroids of path-width at most k has at most $2^{|\mathbb{F}|^{O(k^2)}}$ elements. We can therefore compute, for any integer k and a fixed finite field $\mathbb{F}$, the set of $\mathbb{F}$-representable excluded minors for the class of matroids of path-width k, and this gives as a corollary a polynomial-time algorithm for checking whether the path-width of an $\mathbb{F}$-represented matroid is at most k. We also prove that every excluded pivot-minor for the class of graphs having linear rank-width at most k has at most $2^{2^{O(k^2)}}$ vertices, which also results in a similar algorithmic consequence for linear rank-width of graphs.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases path-width, matroid, linear rank-width, graph, forbidden minor, vertex-minor, pivot-minor

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.40

Related Version *Full Version*: https://arxiv.org/abs/2109.12291

Funding *Mamadou Moustapha Kanté*: Supported by the grant from the French National Research Agency under JCJC program (ASSK: ANR-18-CE40-0025-01).
Eun Jung Kim: Supported by the grant from the French National Research Agency under JCJC program (ASSK: ANR-18-CE40-0025-01).
O-joung Kwon: Supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. NRF-2018R1D1A1B07050294) and by the Institute for Basic Science (IBS-R029-C1).
Sang-il Oum: Supported by the Institute for Basic Science (IBS-R029-C1).

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 40; pp. 40:1–40:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

For a class $\mathcal{C}$ of graphs or matroids, a graph or a matroid is an *excluded minor* for $\mathcal{C}$ if it does not belong to $\mathcal{C}$ but all of its proper minors belong to $\mathcal{C}$.

Robertson and Seymour [20] proved that every minor-closed class of graphs has finitely many excluded minors. This deep theorem has many algorithmic consequences for minor-closed classes of graphs. One of the corollaries is that for each minor-closed class $\mathcal{I}$ of graphs, there exists a monadic second-order formula $\varphi_{\mathcal{I}}$ that expresses the membership in $\mathcal{I}$, as there is a formula to decide whether a graph has a minor isomorphic to a fixed graph. However, the proof of Robertson-Seymour theorem is non-constructive and provides no algorithm of constructing the list of excluded minors and therefore we only know the existence of $\varphi_{\mathcal{I}}$ and do not know how to construct $\varphi_{\mathcal{I}}$ in general.

The class of graphs of path-width at most k is minor-closed and therefore the list of excluded minors for the class of graphs of path-width at most k is finite for each k. Actually, this is also implied by an earlier theorem of Robertson and Seymour [19], stating that graphs of bounded tree-width are well-quasi-ordered under taking minors. But this is still non-constructive. In 1998, Lagergren [14] proved that each excluded minor for the class of graphs of path-width at most k has at most $2^{O(k^4)}$ edges. Therefore we can now construct a monadic second-order formula φ_k to decide whether the path-width of a graph is at most k for each k. Since Courcelle's theorem [3] allows us to decide φ_k on graphs of bounded tree-width in polynomial time, we obtain a polynomial-time algorithm to decide whether an input graph has path-width at most k for each fixed k, even though a direct algorithm was proposed by Bodlaender and Kloks [2].

We aim to prove analogous theorems for the class of matroids of path-width at most k and for the class of graphs of linear rank-width at most k. For a matroid M on the ground set $E(M)$, we define its connectivity function λ_M by

$$\lambda_M(X) = r_M(X) + r_M(E(M) - X) - r(M) \quad \text{for } X \subseteq E(M),$$

where r_M is the rank function of M. The path-width of a matroid M is defined as the minimum *width* of linear orderings of its elements, called *path-decompositions* or *linear layouts*, where the width of a path-decomposition $e_1, e_2, \ldots, e_n$ is defined as the maximum of the values $\lambda_M(\{e_1, e_2, \ldots, e_i\})$ for all $i = 1, 2, \ldots, n$.

For matroid path-width, we do not yet know whether there are only finitely many excluded minors for the class of matroids of path-width at most k. Previously, Koutsonas, Thilikos, and Yamazaki [13] showed a lower bound, proving that the number of excluded minors for the class of matroids of path-width at most k is at least $(k!)^2$. We remark that a class of matroids of bounded path-width is not necessarily well-quasi-ordered under taking minors; Geelen, Gerards, and Whittle [6] showed that there is an infinite antichain of matroids of bounded path-width.

Geelen, Gerards, and Whittle [6] proved that for each finite field $\mathbb{F}$, $\mathbb{F}$-representable matroids of bounded branch-width are well-quasi-ordered under taking minors, as a generalization of the theorem of Robertson and Seymour [19] on graphs of bounded tree-width. This implies that for each finite field $\mathbb{F}$, there are only finitely many $\mathbb{F}$-representable excluded minors for the class of matroids of path-width at most k.

As a corollary, for each finite field $\mathbb{F}$ and an integer k, there exists a monadic second-order formula $\varphi_k^{\mathbb{F}}$ to decide whether an $\mathbb{F}$-representable matroid has path-width at most k, because one can write a monadic second-order formula to describe whether a matroid has a fixed matroid as a minor by Hliněný [7]. Hliněný [7] also proved an analog of Courcelle's theorem for $\mathbb{F}$-represented matroids, showing a fixed-parameter algorithm to decide a monadic second-order formula on $\mathbb{F}$-represented matroids of bounded branch-width, for a finite field $\mathbb{F}$. This allows us to conclude that there "exists" a fixed-parameter tractable algorithm to decide whether an input $\mathbb{F}$-represented matroid has path-width at most k by testing $\varphi_k^{\mathbb{F}}$.

However, the theorem of Geelen, Gerards, and Whittle [6] does not provide any method of constructing the list of $\mathbb{F}$-representable excluded minors and so we did not know how to find $\varphi_k^{\mathbb{F}}$. We are now ready to state our main theorem, showing an explicit upper bound of the size of every $\mathbb{F}$-representable excluded minor.

▶ **Theorem 1.** *For a finite field $\mathbb{F}$ and an integer k, each $\mathbb{F}$-representable excluded minor for the class of matroids of path-width at most k has at most $2^{|\mathbb{F}|^{O(k^2)}}$ elements.*

Thus, by Theorem 1, we "have" an algorithm to construct $\varphi_k^{\mathbb{F}}$ and we "have" a fixed-parameter algorithm to decide whether an input $\mathbb{F}$-represented matroid has path-width at most k. Note that there is a subtle difference between "have" and "there exist"; by Geelen, Gerards, and Whittle [6], we knew that there exists $\varphi_k^{\mathbb{F}}$, but we did not know how to construct it, because their proof is non-constructive. By Theorem 1 we can enumerate all matroids of small size to find the list of all $\mathbb{F}$-representable excluded minors and therefore we can finally construct $\varphi_k^{\mathbb{F}}$.

We remark that Geelen, Gerards, Robertson, and Whittle [5] showed an analogous theorem for branch-width of matroids; for each $k \geq 1$, every excluded minor for the class of matroids of branch-width at most k has at most $(6^{k+1} - 1)/5$ elements.[1]

By extending our method slightly, we also prove a similar theorem for the linear rank-width of graphs as follows.

▶ **Theorem 2.** *Each excluded pivot-minor for the class of graphs of linear rank-width at most k has at most $2^{2^{O(k^2)}}$ vertices.*

Since every vertex-minor obstruction is also a pivot-minor obstruction, we deduce the following.

▶ **Corollary 3.** *Each excluded vertex-minor for the class of graphs of linear rank-width at most k has at most $2^{2^{O(k^2)}}$ vertices.*

The situation is very similar to that of matroids representable over a fixed finite field. Oum [16] showed that graphs of bounded rank-width are well-quasi-ordered under taking pivot-minors, which implies that the list of excluded pivot-minors for the class of graphs of linear rank-width at most k is finite. Again its proof is non-constructive and therefore it provides no algorithm to construct the list. Jeong, Kwon, and Oum [10, 11] proved that any list of excluded pivot-minors characterizing the class of graphs of linear rank-width at most k has at least $2^{\Omega(3^k)}$ graphs.

Corollary 3 answers an open problem of Jeong, Kwon, and Oum [11] on the number of vertices of each excluded vertex-minor for the class of graphs of linear rank-width at most k. Adler, Farley, and Proskurowski [1] characterized excluded vertex-minors for the class of graphs of linear rank-width at most 1. Theorem 6.1 of Kanté and Kwon [12] implies that distance-hereditary excluded vertex-minors for the class of graphs of linear rank-width at most k have at most $O(3^k)$ vertices.

[1] In [5], the connectivity function of matroids is defined to have +1, which makes $(6^k - 1)/5$.

Previously, we only knew the existence of a modulo-2 counting monadic second-order formula Φ_k testing whether a graph has linear rank-width at most k. This is due to the theorem of Courcelle and Oum [4] stating that for each graph H, there is a modulo-2 counting monadic second-order formula to decide whether a graph has a pivot-minor isomorphic to H. As there is a polynomial-time algorithm to decide a modulo-2 counting monadic second-order formula for graphs of bounded rank-width (see [4, Proposition 5.7]), we can conclude that there "exists" a polynomial-time algorithm to decide whether an input graph has linear rank-width at most k. However, this algorithm is based on the existence of Φ_k, and we did not know how to construct Φ_k. Finally, by Theorem 2, we know how to construct Φ_k algorithmically.

Let us now explain the main ideas. We first observe that each excluded minor M has path-width $k+1$, admits a *linked path-decomposition*, which is a path-decomposition satisfying some Menger-like condition, and each proper minor of M has path-width at most k. Secondly, we show that each excluded minor of sufficiently large size has many nested cuts, all of the same value. We finally show that among those cuts of the same value, there are two nested cuts X and Y such that M has a minor on $X \cup (E(M) \setminus Y)$ of path-width $k+1$, contradicting that all proper minors of M have path-width at most k. One of the key ingredients in finding the minor is to use the data structure proposed by Jeong, Kim, and Oum [9]. Based on dynamic programming, they devised fixed-parameter algorithms to decide whether an $\mathbb{F}$-represented matroid has path-width at most k and to decide whether a graph has linear rank-width at most k without using the fact that there are only finitely many excluded minors. Their so-called *B-trajectories* encode partial solutions which may be extended to the full solutions. Here is the idea behind B-trajectories. If $\lambda_M(X) = k$, then the dimension of the vector space spanned by both X and $E(M) \setminus X$ is exactly k. Since the underlying field is finite, this intersection subspace has only finitely many subspaces. Combining this observation with the idea of *typical sequences* appearing in Bodlaender and Kloks [2], Jeong, Kim, and Oum [9] deduce that there are only finitely many collections, called the *full sets*, of meaningful partial solutions (*compact B-trajectories*) at every moment of the dynamic programming algorithm. We indeed prove that among all nested cuts ensured by the large size of M, there are two nested cuts X and Y such that the full set associated with Y can be obtained by applying the same linear transformation to all compact B-trajectories of the full set associated with X, where B is the vector space spanned by both X and $E(M) \setminus X$. The second key ingredient of our proof is the linking theorem for minors of matroids of Tutte [21] and a corresponding theorem for pivot-minors of graphs by Oum [16]; both are analogs of Menger's theorem. These linking theorems will ensure that when two nested cuts display the identical full set up to a certain linear transformation, one can obtain a proper minor or a proper pivot-minor having the same path-width or linear rank-width, respectively.

This paper is organized as follows. Section 2 reviews necessary definitions and known facts on matroids, branch-decompositions, path-decompositions, and Tutte's linking theorem. We review in Section 3 the data structure introduced in Jeong, Kim, and Oum [9]. Section 4 presents a lemma on finding many cuts of the same width inside a *linked* path-decomposition. We present the proof of the main theorem in Section 5. In Section 6, we present the proof for Theorem 2 on linear rank-width of graphs.

2 Preliminaries

For two sets A and B, we write $A \triangle B$ to denote $(A - B) \cup (B - A)$.

2.1 Matroids and minors

A *matroid* is a pair $(E, \mathcal{I})$ of a finite set E and a set $\mathcal{I}$ of subsets of E satisfying the following three properties:

(I1) $\emptyset \in \mathcal{I}$.
(I2) If $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$.
(I3) If $X, Y \in \mathcal{I}$ and $|X| < |Y|$, then there is $e \in Y - X$ such that $X \cup \{e\} \in \mathcal{I}$.

A subset of E is *independent* if it belongs to $\mathcal{I}$. The *ground set* of a matroid $M = (E, \mathcal{I})$ is the set E denoted by $E(M)$. A subset of E is *dependent* if it is not independent.

Let $M = (E, \mathcal{I})$ be a matroid on n elements. We write $\mathcal{I}(M)$ to denote the set of independent sets of a matroid M. A *base* of a matroid is a maximal independent set. A subset of E is *coindependent* if it is disjoint with some base. The *rank* of a set X in a matroid M, denoted by $r_M(X)$, is the size of a maximal independent subset of X in M. The *rank* of a matroid M is $r(M) := r_M(E(M))$. The *connectivity function* of a matroid M, denoted by λ_M is defined as

$$\lambda_M(X) := r_M(X) + r_M(E(M) - X) - r(M)$$

for all $X \subseteq E(M)$. It is easy to verify that λ_M is *submodular*, that is

$$\lambda_M(X) + \lambda_M(Y) \geq \lambda_M(X \cup Y) + \lambda_M(X \cap Y)$$

for all $X, Y \subseteq E(M)$. Also observe that λ_M is *symmetric*, that is $\lambda_M(X) = \lambda_M(E(M) - X)$ for all $X \subseteq E(M)$.

For $X \subseteq E$, the *restriction* $M|_X$ of a matroid M on X is a matroid on the ground set X such that $I \subseteq X$ is an independent set of $M|_X$ if and only if it is an independent set of M. The *deletion* of X from M is the restriction of M on $E - X$, denoted as $M \setminus X$. Another matroid operation is a *contraction*. The contraction of M by X, denoted as M/X, is a matroid with the ground set $E - X$ such that a set $I \subseteq E - X$ is an independent set of M/X if and only if there exists a base B_X of $M|_X$ such that $I \cup B_X$ is an independent set of M. Note that for $Y \subseteq E - X$, $r_{M/X}(Y) = r_M(Y \cup X) - r_M(X)$, where r_M is the rank function of a matroid M. For two matroids M, N, we say that N is a *minor* of M if there exist disjoint subsets C and D of $E(M)$ such that $N = M \setminus D/C$. A minor N of M is *proper* if $E(N) \neq E(M)$.

The following lemma is obtained easily from the above equation on the rank of a minor.

▶ **Lemma 4** (Geelen, Gerards, and Whittle [6, (5.3)]). *Let $M = (E, \mathcal{I})$ be a matroid and let X, C, D be disjoint subsets of E. Then $\lambda_{M \setminus D/C}(X) \leq \lambda_M(X)$. Furthermore, the equality holds if and only if $r_M(X \cup C) = r_M(X) + r_M(C)$ and $r_M(E - X) + r_M(E - D) = r_M(E) + r_M(E - (X \cup D))$.*

2.2 Vector matroids

One of the key examples of matroids is the class of vector matroids. Let A be an $m \times n$ matrix over a field $\mathbb{F}$ whose columns are indexed by a set E of column labels. Then a matroid $M(A)$ on E can be defined from A so that X is independent in $M(A)$ if and only if the corresponding column vectors of A are linearly independent. Such a matroid $M(A)$ is called a *vector matroid* and A is called a *representation* of the matroid $M(A)$. We say that a matroid M is *representable* over $\mathbb{F}$, or equivalently $\mathbb{F}$*-representable* if there is a matrix A over $\mathbb{F}$ such that $M = M(A)$. We say a matroid M is $\mathbb{F}$*-represented* if it is given with its representation over $\mathbb{F}$.

Instead of using matrices, we may regard a vector matroid defined from a finite set of labeled vectors in a vector space, called a *configuration* as in [6]. For a configuration A, we write $M(A)$ to denote the matroid on A such that a subset of A is independent in $M(A)$ if and only if it is linearly independent in the underlying vector space. Note that vectors in a configuration may coincide as we allow two different labels to represent the same vector. We write $\langle A\rangle$ to denote the linear span of the vectors in A.

2.3 Path-width

Let E be a finite set with n elements. A function $f : 2^E \to \mathbb{Z}$ is *submodular* if $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for all $X, Y \subseteq E$ and is *symmetric* if $f(X) = f(E - X)$ for all $X \subseteq E$. We say that a function $f : 2^E \to \mathbb{Z}$ is a *connectivity function* if it is submodular, symmetric, and $f(\emptyset) = 0$.

A *linear layout* of E is a permutation $\sigma = e_1, e_2, \ldots, e_n$ of E. The *width* of a linear layout $\sigma = e_1, e_2, \ldots, e_n$ with respect to f is $\max_{1\leq i<n} f(\{e_1, e_2, \ldots, e_i\})$. The *path-width* of f is the minimum width of all possible linear layouts of E with respect to f.

If f is the matroid connectivity function λ_M of a matroid M, then the linear layout of $E(M)$ is called a *path-decomposition* of M and the path-width of M is defined as the path-width of λ_M.

A linear layout $\sigma = e_1, e_2, \ldots, e_n$ is *linked* if for all $0 \leq i < j \leq n$,

$$\min_{\{e_1,e_2,\ldots,e_i\}\subseteq X\subseteq\{e_1,e_2,\ldots,e_j\}} f(X) = \min_{i\leq \ell\leq j} f(\{e_1, e_2, \ldots, e_\ell\}).$$

Nagamochi [15] presented an algorithm that runs in polynomial time for fixed k to find a linear layout of width at most k if it exists for general connectivity functions. The key step of his algorithm implies the following theorem easily from [15, Lemma 2], which ensures that there always exists a linked linear layout of the optimum width. Actually, his algorithm outputs a linked linear layout.

▶ **Theorem 5** (Nagamochi [15]). *If a connectivity function f has path-width k, then it has a linked linear layout of width at most k.*

2.4 Tutte's linking theorem

▶ **Theorem 6** (Tutte [21]). *Let M be a matroid and A, B be disjoint subsets of $E(M)$. Then*

$$\lambda_M(X) \geq k \text{ for all } A \subseteq X \subseteq E(M) - B$$

if and only if M has a minor N on $A \cup B$ such that $\lambda_N(A) \geq k$.

For a configuration A and $X \subseteq A$, let

$$\partial_A(X) := \langle X\rangle \cap \langle A - X\rangle.$$

Observe that $\lambda_{M(A)}(X) = \dim \partial_A(X)$. The following proposition is essentially due to Geelen, Gerards, and Whittle [6, (5.7)] and we modified their statement with the almost same proof. Note that if $N = M/C \setminus D$ is a minor of M, then we can choose D as a coindependent set in M without changing N, see [18, Lemma 3.3.2]. Thus it is easy to satisfy the requirements of the following proposition from Tutte's linking theorem.

▶ **Proposition 7.** *Let A be a configuration over a field $\mathbb{F}$ and let S, T be subcollections of A such that $S \cap T = \emptyset$. Let C, D be disjoint subcollections of A such that $C \cup D = A - (S \cup T)$, D is coindependent in $M(A)$, and for the minor $N = M(A)/C \setminus D$ of $M(A)$ on $S \cup T$,*

$$\lambda_N(S) = \min_{S \subseteq X \subseteq A-T} \lambda_{M(A)}(X) = k.$$

Then for all subcollections Z of A, if $S \subseteq Z \subseteq A - T$ and $\lambda_{M(A)}(Z) = k$, then the following hold.

- **(i)** *For all $x, y \in \langle Z \rangle$, $x - y \in \langle C \rangle$ if and only if $x - y \in \langle C \cap Z \rangle$.*
- **(ii)** *For all $x, y \in \langle A - Z \rangle$, $x - y \in \langle C \rangle$ if and only if $x - y \in \langle C - Z \rangle$.*
- **(iii)** *For all $x, y \in \partial_A(Z)$, $x - y \in \langle C \rangle$ if and only if $x = y$.*
- **(iv)** *If Z' is also a subcollection of A such that $S \subseteq Z' \subseteq A - T$ and $\lambda_{M(A)}(Z') = k$, then for each $x \in \partial_A(Z')$, there is a unique $y \in \partial_A(Z)$ such that $x - y \in \langle C \rangle$. Moreover, $x - y \in \langle C \cap (Z \triangle Z') \rangle$.*

Proof. Let $M = M(A)$. Since D is coindependent, $r_M(A - D) = r_M(A)$. Let $C_1 = C \cap Z$, $D_1 = D \cap Z$, $C_2 = C - Z$, and $D_2 = D - Z$. By Lemma 4,

$$r_M(A - Z) + r_M(A - D_2) = r_M(A) + r_M(A - (Z \cup D_2)),$$
$$r_M(Z \cup C_2) = r_M(Z) + r_M(C_2).$$

As $r_M(A - D_2) = r_M(A)$, from the first equation, we have $r_M(A - Z) = r_M(A - (Z \cup D_2)) = r_M(T \cup C_2)$ and so

$$\langle A - Z \rangle = \langle T \cup C_2 \rangle. \tag{1}$$

From the second equation, we have

$$\langle Z \rangle \cap \langle C_2 \rangle = \{0\}. \tag{2}$$

By symmetry between S and T and between Z and $V - Z$, we have

$$\langle Z \rangle = \langle S \cup C_1 \rangle \text{ and } \langle A - Z \rangle \cap \langle C_1 \rangle = \{0\}. \tag{3}$$

Suppose that $x, y \in \langle Z \rangle$ and $x - y \in \langle C \rangle$. Let $c_1 \in \langle C_1 \rangle$ and $c_2 \in \langle C_2 \rangle$ such that $x - y = c_1 + c_2$. Then $x - y - c_1 \in \langle C_2 \rangle \cap \langle Z \rangle$. By (2), $x - y - c_1 = 0$ and so $x - y \in \langle C_1 \rangle$. This proves (i). By symmetry, (ii) is also proved.

By (i) and (ii), if $x, y \in \partial_A(Z)$ and $x - y \in \langle C \rangle$, then $x - y \in \langle C \cap Z \rangle \cap \langle C - Z \rangle$. By (2), $\langle C \cap Z \rangle \cap \langle C - Z \rangle = \{0\}$ and therefore $x = y$. This proves (iii).

To prove (iv), suppose that $x \in \partial_A(Z')$. By (1) applied to Z', there exist $t \in \langle T \rangle$ and $c_2 \in \langle C - Z' \rangle$ such that $x = t + c_2$. Similarly, by (3), there exist $s \in \langle S \rangle$ and $c_1 \in \langle C \cap Z' \rangle$ such that $x = s + c_1$. We can write $c_1 = c_{11} + c_{12}$ for $c_{11} \in \langle C \cap (Z \cap Z') \rangle$ and $c_{12} \in \langle C \cap (Z' - Z) \rangle$ and write $c_2 = c_{21} + c_{22}$ for $c_{21} \in \langle C \cap (Z - Z') \rangle$ and $c_{22} \in \langle C - (Z \cup Z') \rangle$. Let us define $y = s + c_{11} - c_{21} = t + c_{22} - c_{12}$. Then $y \in \partial_A(Z)$ because $s + c_{11} - c_{21} \in \langle Z \rangle$ and $t + c_{22} - c_{12} \in \langle A - Z \rangle$. Now observe that $x - y = c_{12} + c_{21} \in \langle C \cap (Z \triangle Z') \rangle$. This proves that the desired y exists. By (iii), such y is unique. ◀

3 Full sets

We review the concepts of B-trajectories and full sets introduced by Jeong, Kim, and Oum [9].

3.1 B-trajectories

Let B be a vector space. A *statistic* is a triple $a = (L, R, \lambda)$ of subspaces L, R of B and a non-negative integer λ. For convenience, we write $L(a) = L$, $R(a) = R$, and $\lambda(a) = \lambda$. A *B-trajectory* is a sequence $\Gamma = a_0, a_1, \ldots, a_n$ of statistics for a non-negative integer n such that

- $R(a_0) = L(a_n)$,
- $L(a_0) \subseteq L(a_1) \subseteq \cdots \subseteq L(a_n) \subseteq B$,
- $R(a_n) \subseteq R(a_{n-1}) \subseteq \cdots \subseteq R(a_0) \subseteq B$.

The width of Γ is $\max_{0\leq i\leq n} \lambda(a_i)$. We write $\Gamma(i)$ to denote a_i. The *length* of Γ, denoted by $|\Gamma|$, is $n+1$.

Let $A = \{e_1, e_2, \ldots, e_n\}$ be a configuration over a field $\mathbb{F}$. From a path-decomposition $\sigma = e_1, e_2, \ldots, e_n$ of a represented matroid $M = M(A)$, we can obtain its *canonical B-trajectory* as follows. For $i = 0, 1, 2, \ldots, n$, let

$$\begin{aligned} L_i &= \langle e_1, e_2, \ldots, e_i\rangle \cap B, \\ R_i &= \langle e_{i+1}, e_{i+2}, \ldots, e_n\rangle \cap B, \text{ and} \\ \lambda_i &= \dim\langle e_1, e_2, \ldots, e_i\rangle \cap \langle e_{i+1}, e_{i+2}, \ldots, e_n\rangle - \dim L_i \cap R_i. \end{aligned}$$

Note that $L_0 = R_n = \{0\}$ and $\lambda_0 = \lambda_n = 0$. Let $a_i = (L_i, R_i, \lambda_i)$ for $i = 0, 1, 2, \ldots, n$. Then it is easy to see that $\Gamma = a_0, a_1, a_2, \ldots, a_n$ is a B-trajectory, which we call the *canonical B-trajectory* of σ. If Γ is a canonical B-trajectory of some path-decomposition σ of $M = M(A)$, then we say Γ is *realizable* in A.

For a B-trajectory $\Gamma = a_0, a_1, a_2, \ldots, a_n$, the *compactification* of Γ, denoted by $\tau(\Gamma)$, is a B-trajectory obtained from Γ by applying the following operations repeatedly until no further operations can be applied.

- Remove an entry a_i if $a_{i-1} = a_i$.
- Remove a subsequence $a_{i+1}, a_{i+2}, \ldots, a_{j-1}$ if $i+1 < j$, $L(a_i) = L(a_j)$, $R(a_i) = R(a_j)$, and either $\lambda(a_i) \leq \lambda(a_k) \leq \lambda(a_j)$ for all $k \in \{i+1, i+2, \ldots, j-1\}$ or $\lambda(a_i) \geq \lambda(a_k) \geq \lambda(a_j)$ for all $k \in \{i+1, i+2, \ldots, j-1\}$.

We say that a B-trajectory is *compact* if $\tau(\Gamma) = \Gamma$. Let $U_k(B)$ be the set of all compact B-trajectories of width at most k.

▶ **Lemma 8** (Jeong, Kim, and Oum [9, Lemma 11]). *Let B be a vector space over a finite field $\mathbb{F}$ with dimension θ. Then*

$$|U_k(B)| \leq 2^{9\theta+2}|\mathbb{F}|^{\theta(\theta-1)}2^{2(2\theta+1)k}.$$

We can define binary relations which compare two B-trajectories as follows [9]. For two statistics a and b, we write $a \leq b$ if

$$L(a) = L(b), \ R(a) = R(b), \text{ and } \lambda(a) \leq \lambda(b).$$

For two B-trajectories Γ_1 and Γ_2, we write $\Gamma_1 \leq \Gamma_2$ if the lengths of Γ_1 and Γ_2 are the same, say n, and $\Gamma_1(i) \leq \Gamma_2(i)$ for all $0 \leq i \leq n-1$. A B-trajectory Γ^* is called an *extension* of a B-trajectory Γ if Γ^* can be obtained by repeating some statistics of Γ. We say that $\Gamma_1 \preccurlyeq \Gamma_2$ if there are extensions Γ_1^* of Γ_1 and Γ_2^* of Γ_2 such that $\Gamma_1^* \leq \Gamma_2^*$.

3.2 A full set

We review the *full set* notion introduced by Jeong, Kim, and Oum [9] used for their algorithm to decide the path-width of represented matroids. Let A be a configuration of vectors in a vector space V over a field $\mathbb{F}$. Let B be a subspace of V.

The *full set* of A of width k with respect to B, denoted by $\mathrm{FS}_k(A, B)$, is the set of all compact B-trajectories Γ of width at most k such that there exists a B-trajectory Δ realizable in A with $\Delta \preccurlyeq \Gamma$. From the definition, it is clear that

$$\mathrm{FS}_k(A, \{0\}) \neq \emptyset \text{ if and only if } M(A) \text{ has path-width at most } k.$$

By Lemma 8, the number of B-trajectories in $\mathrm{FS}_k(A, B)$ is bounded by a function of $|\mathbb{F}|$, $\dim B$, and k.

The following lemma is an immediate consequence of Jeong, Kim, and Oum [9, Propositions 35 and 36].

▶ **Lemma 9.** *Let A, A' be configurations in a vector space V. Let k be a non-negative integer. Let B be a subspace of V. If $\mathrm{FS}_k(A, B) = \mathrm{FS}_k(A', B)$, then $\mathrm{FS}_k(A, \{0\}) = \mathrm{FS}_k(A', \{0\})$.*

▶ **Lemma 10.** *Let A_1, A_1', A_2, A_2' be configurations in a vector space V. Let k be a non-negative integer. Let B be a subspace of V such that $(\langle A_1\rangle + B) \cap (\langle A_2\rangle + B) = B$ and $(\langle A_1'\rangle + B) \cap (\langle A_2'\rangle + B) = B$. If $\mathrm{FS}_k(A_1, B) = \mathrm{FS}_k(A_1', B)$ and $\mathrm{FS}_k(A_2, B) = \mathrm{FS}_k(A_2', B)$, then $\mathrm{FS}_k(A_1 \cup A_2, B) = \mathrm{FS}_k(A_1' \cup A_2', B)$.*

For a configuration $A = \{e_1, e_2, \ldots, e_n\}$ and a linear transformation ϕ, we write $\phi(A)$ to denote a configuration $\{\phi(e_1), \phi(e_2), \ldots, \phi(e_n)\}$.

If B_1 and B_2 are subspaces of the same dimension and ϕ is a bijective linear transformation from B_1 to B_2, then for each B_1-trajectory Γ we can define a B_2-trajectory $\Delta := \phi(\Gamma)$ in the following way:

$$L(\Delta(i)) = \phi(L(\Gamma(i))), \quad R(\Delta(i)) = \phi(R(\Gamma(i))), \quad \lambda(\Delta(i)) = \lambda(\Gamma(i)),$$

for every $0 \leq i \leq |\Gamma| - 1$. For a set of B-trajectories $\mathcal{R}$, we define the set $\phi(\mathcal{R}) = \{\phi(\Gamma) : \Gamma \in \mathcal{R}\}$.

Observe that if ϕ is a linear transformation on $\langle A\rangle$ that is injective on $\langle A_1\rangle$ and B_1 is a subspace of $\langle A_1\rangle$, then

$$\phi(\mathrm{FS}_k(A_1, B_1)) = \mathrm{FS}_k(\phi(A_1), \phi(B_1)).$$

Here on the right-hand side, we use ϕ values for all vectors in $\langle A_1\rangle$ but on the left-hand side, we only use ϕ for vectors in B_1.

We can deduce the following lemma easily from Lemmas 9 and 10. We omit its proof.

▶ **Lemma 11.** *Let k be a non-negative integer and let $\mathbb{F}$ be a field. Let A be a configuration in a vector space V over $\mathbb{F}$ and let A' be a configuration in a vector space V' over $\mathbb{F}$. Let (A_1, A_2) be a partition of A and (A_1', A_2') be a partition of A'. If there is a bijective linear transformation $\phi : \partial_A(A_1) \to \partial_{A'}(A_1')$ such that*

$$\phi(\mathrm{FS}_k(A_1, \partial_A(A_1))) = \mathrm{FS}_k(A_1', \partial_{A'}(A_1')) \textit{ and}$$
$$\phi(\mathrm{FS}_k(A_2, \partial_A(A_1))) = \mathrm{FS}_k(A_2', \partial_{A'}(A_1')),$$

then the path-width of $M(A)$ is at most k if and only if the path-width of $M(A')$ is at most k.

4 Finding many repeated cuts

The following lemma finds many cuts in the linked path-decomposition that are of the same width and linked each other.

▶ **Lemma 12.** *Let $\ell \geq 4$ be an integer. Let $a_0, a_1, a_2, \ldots, a_n$ be a sequence of integers such that $a_i \geq a_0 = a_n$ for all $0 \leq i \leq n$ and $|a_i - a_{i+1}| \leq 1$. If*

$$n \geq \left(\ell - 1 + \frac{2(\ell-2)}{\ell-3}\right)(\ell-2)^{\max_{0\leq i\leq n}(a_i - a_0)} - \frac{2(\ell-2)}{\ell-3},$$

then there exist $0 \leq i_1 < i_2 < i_3 < \cdots < i_\ell \leq n$ and w such that

$$a_{i_1} = a_{i_2} = \cdots = a_{i_\ell} = w \textit{ and } a_i \geq w \textit{ for all } i_1 \leq i \leq i_\ell.$$

Proof. We proceed by induction on $M = \max_{0\leq i\leq n}(a_i - a_0)$. It is trivial if $M = 0$. Let $m = |\{i \in \{0, 1, \ldots, n\} : a_i = a_0\}|$. If $m \geq \ell$, then we are done. Thus we may assume that $m \leq \ell - 1$. Then there exists a subsequence $a_p, a_{p+1}, \ldots, a_q$ such that $a_i > a_0$ for all $p \leq i \leq q$, and $q - p + 1 \geq \frac{n}{m-1} - 1 \geq \frac{n}{\ell-2} - 1$. Equivalently, $q - p + \frac{2(\ell-2)}{\ell-3} \geq \frac{1}{\ell-2}\left(n + \frac{2(\ell-2)}{\ell-3}\right)$ and therefore

$$q - p \geq \left(\ell - 1 + \frac{2(\ell-2)}{\ell-3}\right)(\ell-2)^{M-1} - \frac{2(\ell-2)}{\ell-3}.$$

We may assume that $q - p$ is chosen as a maximum. Then by the assumption that $|a_i - a_{i+1}| \leq 1$, we deduce that $a_p = a_q = a_0 + 1$. Now we apply the induction hypothesis to the subsequence $a_p, a_{p+1}, \ldots, a_q$ to conclude the proof. ◀

We will apply Lemma 12 to a sequence $a_0, a_1, a_2, \ldots, a_n$ obtained from a linked path-decomposition $\sigma = e_1, e_2, \ldots, e_n$, where $a_i = \lambda_M(\{e_1, e_2, \ldots, e_i\})$ for $i = 0, 1, 2, \ldots, n$. It is easy to verify that any path-decomposition σ of a represented matroid meets the requirement that $|a_i - a_{i+1}| \leq 1$ of Lemma 12. The next lemma is needed.

▶ **Lemma 13.** *Let M be a matroid. If $e \in X \subseteq E(M)$, then $|\lambda_M(X) - \lambda_M(X - \{e\})| \leq 1$.*

Proof. By the submodularity of the connectivity function, we have $\lambda_M(X - \{e\}) + \lambda_M(\{e\}) \geq \lambda_M(X)$. Since $\lambda_M(\{e\}) \leq 1$, we have $\lambda_M(X) \leq \lambda_M(X - \{e\}) + 1$. Since λ_M is symmetric, we deduce that $\lambda_M(X - \{e\}) \leq \lambda_M(X) + 1$. ◀

5 The proof

The following proposition proves Theorem 1.

▶ **Proposition 14.** *Let $\mathbb{F}$ be a finite field and k be a non-negative integer. Let M be an $\mathbb{F}$-representable matroid of path-width larger than k. Let $\ell = 2^{2^{9k+11}|\mathbb{F}|^{k(k+1)}2^{2(2k+3)k}} + 1$. If*

$$|E(M)| \geq \left(\ell - 1 + \frac{2(\ell-2)}{\ell-3}\right)(\ell-2)^{k+1} - \frac{2(\ell-2)}{\ell-3},$$

then there is $e \in E(M)$ such that M/e or $M \setminus e$ has path-width larger than k.

Proof. Let A be a configuration in a vector space over $\mathbb{F}$ such that $M = M(A)$. We may assume that $M \setminus e$ and M/e has path-width at most k for every $e \in E(M)$. This implies that M has path-width exactly $k + 1$ and by Theorem 5, there is a linked path-decomposition $\sigma = e_1, e_2, \ldots, e_n$ of M of width $k + 1$. We identify e_i with a vector in A.

For $i = 0, 1, 2, \ldots, n$, let $a_i = \lambda_M(\{e_1, e_2, \ldots, e_i\})$. Then $0 \le a_i \le k+1$ for all i.

By Lemma 12, there exist integers $0 \le t_1 < t_2 < \cdots < t_\ell \le n$ and $0 \le \theta \le k+1$ such that $a_{t_1} = a_{t_2} = \cdots = a_{t_\ell} = \theta$ and $a_i \ge \theta$ for all $t_1 \le i \le t_\ell$. Let $A_i = \{e_1, e_2, \ldots, e_{t_i}\}$ and $B_i = \partial_A(A_i)$ for $1 \le i \le \ell$.

Since σ is a linked path-decomposition, $\lambda_M(X) \ge \theta$ for all $A_1 \subseteq X \subseteq A_\ell$. By Theorem 6, there are disjoint subcollections C, D of A such that $C \cup D = A - (A_1 \cup (A - A_\ell))$ and $\lambda_{M/C \setminus D}(A_1) = \theta$. We may assume that D is coindependent, see [18, Lemma 3.3.2]. Let $\pi : \langle A \rangle \to \langle A \rangle / \langle C \rangle$ be the linear transformation mapping $x \in \langle A \rangle$ to an equivalence class $[x]$ containing x where two vectors x and x' are equivalent if and only if $x - x' \in \langle C \rangle$. Let $B = \pi(\partial_A(A_1))$.

By (iii) and (iv) of Proposition 7, $\dim B = \theta$ and $\pi(\partial_A(A_i)) = \pi(\partial_A(A_j))$ for all $1 \le i < j \le \ell$.

Observe that $\pi(\mathrm{FS}_k(A_i, \partial_A(A_i))) \subseteq U_k(B)$. Since ℓ is big enough, by Lemma 8 and the pigeon-hole principle, there exist $1 \le i < j \le \ell$ such that $\pi(\mathrm{FS}_k(A_i, \partial_A(A_i))) = \pi(\mathrm{FS}_k(A_j, \partial_A(A_j)))$.

Let $C' = C \cap (A_j - A_i)$ and $D' = D \cap (A_j - A_i)$. Let $\phi : \langle A \rangle \to \langle A \rangle / \langle C' \rangle$ be the linear transformation mapping $x \in \langle A \rangle$ to an equivalence class containing x where two elements x, y are equivalent if and only if $x - y \in \langle C' \rangle$.

Let $B' = \phi(\partial_A(A_i))$. Since $C' \subseteq C$, by (iii) of Proposition 7, we have $\dim B' = \theta$. Furthermore, from (iv) of Proposition 7, we deduce that for $x \in \partial_A(A_i)$ and $y \in \partial_A(A_j)$, $\pi(x) = \pi(y)$ if and only if $\phi(x) = \phi(y)$. Therefore, $B' = \phi(\partial_A(A_j))$ and $\phi(\mathrm{FS}_k(A_i, \partial_A(A_i))) = \phi(\mathrm{FS}_k(A_j, \partial_A(A_j)))$.

We claim that ϕ is an injection on $\langle A_i \rangle$. Suppose that $x, y \in \langle A_i \rangle$ and $x - y \in \langle C' \rangle = \langle C \cap (A_j - A_i) \rangle \subseteq \langle A - A_i \rangle$. Then $x - y \in \langle C \rangle$ and by (i) of Proposition 7, we deduce that $x - y \in \langle C \cap A_i \rangle \subseteq \langle A_i \rangle$. This would imply that $x - y \in \partial_A(A_i)$ and therefore $x = y$ by (iii) of Proposition 7. By symmetry, we can also deduce that ϕ is an injection on $\langle A - A_j \rangle$.

Let $N = M(A)/C' \setminus D'$. Then $A' = \phi(A_i \cup (A - A_j))$ is a configuration in the vector space $\langle A \rangle / \langle C' \rangle$ such that $N = M(A')$. Since $B' \subseteq \langle \phi(A_i) \rangle$ and $B' \subseteq \langle \phi(A - A_j) \rangle$, we have $B' \subseteq \partial_{A'}(\phi(A_i))$. By Lemma 4, $\dim \partial_{A'}(\phi(A_i)) \le \theta$ and therefore $B' = \partial_{A'}(\phi(A_i))$.

Since ϕ is an injection on A_i, $\mathrm{FS}_k(\phi(A_i), \partial_{A'}(\phi(A_i))) = \phi(\mathrm{FS}_k(A_j), \partial_A(A_j))$. Since ϕ is an injection on $A - A_j$, trivially $\mathrm{FS}_k(\phi(A - A_j), \partial_{A'}(\phi(A - A_j))) = \phi(\mathrm{FS}_k(A - A_j), \partial_A(A - A_j))$.

Since N is a proper minor of M, the path-width of N is at most k. By Lemma 11, M has path-width at most k if and only if N has path-width at most k and therefore we deduce that the path-width of M is at most k, contradicting the assumption. ◀

6 Obstructions to linear rank-width

All graphs in this section are simple, having no loops and no parallel edges.

For a graph G, the *cut-rank* function ρ_G of G is defined as a function that maps a set X of vertices of G to the rank of the $X \times (V(G) - X)$ matrix over the binary field whose ab-entry is 1 if and only if $a \in X$ is adjacent to $b \in V(G) - X$. It is known that ρ_G is symmetric and submodular, see Oum and Seymour [17], and therefore it is a connectivity function. We remark that $\rho_G(\emptyset) = \rho_G(V(G)) = 0$. The *linear rank-width* of a graph G is defined to be the path-width of ρ_G.

For a pair (x, y) of distinct vertices of a graph G, *flipping* (x, y) is an operation that adds an edge xy if x, y are non-adjacent in G and deletes the edge xy otherwise. For an edge uv of a graph G, we write $G \wedge uv$ to denote the graph G' on $V(G)$ obtained by the following procedures.

1. For every pair $x \in N(u) \cap N(v)$ and $y \in N(u) - N(v)$, flip (x, y).
2. For every pair $x \in N(u) \cap N(v)$ and $y \in N(v) - N(u)$, flip (x, y).
3. For every pair $x \in N(u) - N(v)$ and $y \in N(v) - N(u)$, flip (x, y).
4. Swap the label of u and v.

This operation is called the *pivot.* We remark that the purpose of the last operation is to make $G \wedge uv \wedge vw = G \wedge uw$, see Oum [16]. Here is an important property of pivots with respect to the cut-rank function.

▶ **Proposition 15** (See Oum [16]). *If $H = G \wedge uv$, then $\rho_H(X) = \rho_G(X)$ for all $X \subseteq V(G)$.*

We say that a graph H is a *pivot-minor* of a graph G if H is an induced subgraph of a graph obtained from G by applying some sequence of pivots. We say that a pivot-minor H of G is *proper* if $V(H) \neq V(G)$. Since deleting a vertex never increases the cut-rank function, we deduce the following easily from the previous proposition.

▶ **Corollary 16.** *If H is a pivot-minor of G, then the linear rank-width of H is at most the linear rank-width of G.*

Oum [16] proved an analog of Tutte's linking theorem for pivot-minors.

▶ **Theorem 17.** *Let G be a graph and let S, T be disjoint vertex sets of G. Then there exists a pivot-minor H on $S \cup T$ such that $\rho_H(S) = \min_{S \subseteq X \subseteq V(G)-T} \rho_G(X)$.*

Let us now show how to represent a graph with a *subspace arrangement.* A *subspace arrangement* $\mathcal{V}$ over a field $\mathbb{F}$ is a finite set of subspaces of a finite-dimensional vector space over $\mathbb{F}$. We usually write a subspace arrangement as a family $\mathcal{V} = \{V_i\}_{i \in E}$ of subspaces indexed by a finite set E.

A *linear layout* of a subspace arrangement $\mathcal{V}$ is a permutation $\sigma = V_1, V_2, \dots, V_n$ of $\mathcal{V}$. The *width* of a linear layout $\sigma = V_1, V_2, \dots, V_n$ is equal to

$$\max_{1 \leq i < n} \dim(V_1 + V_2 + \cdots + V_i) \cap (V_{i+1} + V_{i+2} + \cdots + V_n).$$

Note that this function is a connectivity function on $\mathcal{V}$. The *path-width* of $\mathcal{V}$ is the minimum width of linear layouts of $\mathcal{V}$. If $|\mathcal{V}| \leq 1$, then we define the width of its linear layout to be 0 and its path-width to be 0.

As observed in [9, Section VII], for a matroid M represented by a configuration A, if we take $\mathcal{V} = \{\langle v \rangle : v \in A\}$, then the path-width of $\mathcal{V}$ is equal to the path-width of $M(A)$.

We are now going to review the construction, appeared in [9, Section VIII], of a subspace arrangement from graphs to relay the concept of linear rank-width to the path-width of its corresponding subspace arrangement. For a graph G on the vertex set $\{1, 2, \dots, n\}$, let us define a subspace arrangement over the binary field as follows. Let $\{e_1, e_2, \dots, e_n\}$ be the standard basis of $\mathbb{F}_2^n$ where $\mathbb{F}_2$ is the binary field. Let $v_i = \sum_{j \in N_G(i)} e_j$, where $N_G(i)$ denotes the set of neighbors of i. Let $V_i = \langle e_i, v_i \rangle$ and let $\mathcal{V}_G = \{V_i\}_{i \in V(G)}$.

Here is the key observation.

▶ **Lemma 18** (Jeong, Kim, and Oum [9, Lemma 52]). *For $X \subseteq V(G)$,*

$$\dim\big((\sum_{i \in X} V_i) \cap (\sum_{j \in V(G)-X} V_j)\big) = 2\rho_G(X).$$

▶ **Corollary 19.** *The path-width of $\mathcal{V}_G$ is equal to twice the linear rank-width of G.*

For a subset X of $V(G)$, let $I_X = \{e_i : i \in X\}$, $A_X = \{v_i : i \in X\}$, and $\partial_X = \langle I_X \cup A_X\rangle \cap \langle I_{V(G)-X} \cup A_{V(G)-X}\rangle$. By Lemma 18, $\dim \partial_X = 2\rho_G(X)$. One can see that I_Z is a set of some column vectors in the $n \times n$ identity matrix and A_Z is a set of some column vectors in the adjacency matrix of G. Let M_G be the binary matroid represented by the matrix $(I_n \; A(G))$, where I_n is the $n \times n$ identity matrix and $A(G)$ is the adjacency matrix of G.

In Subsection 3.2, we reviewed the concept of full sets for the context of represented matroids or configurations. In fact, Jeong, Kim, and Oum [9] introduced full sets in more general form for subspace arrangements.

Here we are going to show the difference compared to Subsections 3.1 and 3.2. For a subspace arrangement $\mathcal{V}$ and its linear layout $\sigma = V_1, V_2, \ldots, V_n$, the *canonical B-trajectory* is defined as follows. For $i = 0, 1, \ldots, n$, let $L_i = (\sum_{j=1}^{i} V_j) \cap B$, $R_i = (\sum_{j=i+1}^{n} V_j) \cap B$, $\lambda_i = \dim(\sum_{j=1}^{i} V_j) \cap (\sum_{j=i+1}^{n} V_j) - \dim L_i \cap R_i$, and $a_i = (L_i, R_i, \lambda_i)$. Then $\Gamma = a_0, a_1, a_2, \ldots, a_n$ is the *canonical B-trajectory* of σ. We say that Γ is realizable in $\mathcal{V}$ if it is a canonical B-trajectory of some linear layout of $\mathcal{V}$.

For a subspace arrangement $\mathcal{V}$, $\mathrm{FS}_k(\mathcal{V}, B)$ is defined as the set of all compact B-trajectories Γ of width at most k such that there exists a B-trajectory Δ realizable in $\mathcal{V}$ with $\Delta \preceq \Gamma$.

Lemmas 9 and 10 are special cases of the following two lemmas easily deduced from the result of Jeong, Kim, and Oum [9].

▶ **Lemma 20.** *Let $\mathcal{V}$, $\mathcal{V}'$ be subspace arrangements over a field $\mathbb{F}$. Let k be a non-negative integer. Let B be a subspace of $\langle \mathcal{V} \cup \mathcal{V}'\rangle$. If $\mathrm{FS}_k(\mathcal{V}, B) = \mathrm{FS}_k(\mathcal{V}', B)$, then $\mathrm{FS}_k(\mathcal{V}, \{0\}) = \mathrm{FS}_k(\mathcal{V}', \{0\})$.*

▶ **Lemma 21.** *Let $\mathcal{V}_1$, $\mathcal{V}_1'$, $\mathcal{V}_2$, $\mathcal{V}_2'$ be subspace arrangements over a field $\mathbb{F}$. Let k be a non-negative integer. Let B be a subspace of $\langle \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_1' \cup \mathcal{V}_2'\rangle$ such that $(\langle \mathcal{V}_1\rangle + B) \cap (\langle \mathcal{V}_2\rangle + B) = B$ and $(\langle \mathcal{V}_1'\rangle + B) \cap (\langle \mathcal{V}_2'\rangle + B) = B$. If $\mathrm{FS}_k(\mathcal{V}_1, B) = \mathrm{FS}_k(\mathcal{V}_1', B)$ and $\mathrm{FS}_k(\mathcal{V}_2, B) = \mathrm{FS}_k(\mathcal{V}_2', B)$, then $\mathrm{FS}_k(\mathcal{V}_1 \cup \mathcal{V}_2, B) = \mathrm{FS}_k(\mathcal{V}_1' \cup \mathcal{V}_2', B)$.*

We can deduce the following lemma easily from Lemmas 20 and 21 by the same method of deducing Lemma 10 from Lemmas 9 and 10.

▶ **Lemma 22.** *Let k be a non-negative integer and let $\mathbb{F}$ be a field. Let $\mathcal{V}$ be a subspace arrangement over $\mathbb{F}$ and let $\mathcal{V}'$ be a subspace arrangement over $\mathbb{F}$. Let $(\mathcal{V}_1, \mathcal{V}_2)$ be a partition of $\mathcal{V}$ and $(\mathcal{V}_1', \mathcal{V}_2')$ be a partition of $\mathcal{V}'$. If there is a bijective linear transformation $\phi : \partial_{\mathcal{V}}(\mathcal{V}_1) \to \partial_{\mathcal{V}'}(\mathcal{V}_1')$ such that*

$$\phi(\mathrm{FS}_k(\mathcal{V}_1, \partial_{\mathcal{V}}(\mathcal{V}_1))) = \mathrm{FS}_k(\mathcal{V}_1', \partial_{\mathcal{V}'}(\mathcal{V}_1')) \text{ and } \phi(\mathrm{FS}_k(\mathcal{V}_2, \partial_{\mathcal{V}}(\mathcal{V}_1))) = \mathrm{FS}_k(\mathcal{V}_2', \partial_{\mathcal{V}'}(\mathcal{V}_1')),$$

then the path-width of $\mathcal{V}$ is at most k if and only if the path-width of $\mathcal{V}'$ is at most k.

The following proposition implies Theorem 2 and Corollary 3. We omit its proof.

▶ **Proposition 23.** *Let G be a graph of linear rank-width larger than k. Let $\ell = 2^{2^{18(k+1)+2+(2k+2)(2k+1)+2(4k+3)2k}} + 1$. If G has more than*

$$\left(\ell - 1 + \frac{2(\ell-2)}{\ell-3}\right)(\ell-2)^{k+1} - \frac{2(\ell-2)}{\ell-3},$$

vertices, then G has a proper pivot-minor H whose linear rank-width is larger than k.

References

1 Isolde Adler, Arthur M. Farley, and Andrzej Proskurowski. Obstructions for linear rank-width at most 1. *Discrete Appl. Math.*, 168:3–13, 2014. doi:10.1016/j.dam.2013.05.001.

2 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996. doi:10.1006/jagm.1996.0049.

3 Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

4 Bruno Courcelle and Sang-il Oum. Vertex-minors, monadic second-order logic, and a conjecture by Seese. *J. Combin. Theory Ser. B*, 97(1):91–126, 2007. doi:10.1016/j.jctb.2006.04.003.

5 James F. Geelen, A. M. H. Gerards, Neil Robertson, and Geoff Whittle. On the excluded minors for the matroids of branch-width k. *J. Combin. Theory Ser. B*, 88(2):261–265, 2003. doi:10.1016/S0095-8956(02)00046-1.

6 James F. Geelen, A. M. H. Gerards, and Geoff Whittle. Branch-width and well-quasi-ordering in matroids and graphs. *J. Combin. Theory Ser. B*, 84(2):270–290, 2002. doi:10.1006/jctb.2001.2082.

7 Petr Hliněný. The Tutte polynomial for matroids of bounded branch-width. *Combin. Probab. Comput.*, 15(3):397–409, 2006. doi:10.1017/S0963548305007297.

8 Jisu Jeong, Eun Jung Kim, and Sang-il Oum. Constructive algorithm for path-width of matroids. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1695–1704, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611974331.ch116.

9 Jisu Jeong, Eun Jung Kim, and Sang-il Oum. The "art of trellis decoding" is fixed-parameter tractable. *IEEE Trans. Inform. Theory*, 63(11):7178–7205, 2017. An extended abstract appeared in a conference proceeding [8]. doi:10.1109/TIT.2017.2740283.

10 Jisu Jeong, O-joung Kwon, and Sang-il Oum. Excluded vertex-minors for graphs of linear rank-width at most k. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 221–232, Kiel, Germany, 2013. Schloss Dagstuhl. Leibniz-Zent. Inform. doi:10.4230/LIPIcs.STACS.2013.221.

11 Jisu Jeong, O-joung Kwon, and Sang-il Oum. Excluded vertex-minors for graphs of linear rank-width at most k. *European J. Combin.*, 41:242–257, 2014. doi:10.1016/j.ejc.2014.04.010.

12 Mamadou Moustapha Kanté and O-joung Kwon. Linear rank-width of distance-hereditary graphs II. Vertex-minor obstructions. *European J. Combin.*, 74:110–139, 2018. doi:10.1016/j.ejc.2018.07.009.

13 Athanassios Koutsonas, Dimitrios M. Thilikos, and Koichi Yamazaki. Outerplanar obstructions for matroid pathwidth. *Discrete Math.*, 315–316:95–101, 2014. doi:10.1016/j.disc.2013.10.007.

14 Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Combin. Theory Ser. B*, 73(1):7–40, 1998. doi:10.1006/jctb.1997.1788.

15 Hiroshi Nagamochi. Linear layouts in submodular systems. In Kun-Mao Chao, Tsan-Sheng Hsu, and Der-Tsai Lee, editors, *ISAAC '12*, volume 7676 of *Lecture Notes in Comput. Sci.*, pages 475–484. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-35261-4_50.

16 Sang-il Oum. Rank-width and vertex-minors. *J. Combin. Theory Ser. B*, 95(1):79–100, 2005. doi:10.1016/j.jctb.2005.03.003.

17 Sang-il Oum and Paul Seymour. Approximating clique-width and banch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

18 James Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, second edition, 2011.

19 Neil Robertson and Paul Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Combin. Theory Ser. B*, 48(2):227–254, 1990. doi:10.1016/0095-8956(90)90120-O.

20 Neil Robertson and Paul Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

21 William T. Tutte. Menger's theorem for matroids. *J. Res. Nat. Bur. Standards Sect. B*, 69B:49–53, 1965.

Fairly Popular Matchings and Optimality

Telikepalli Kavitha
Tata Institute of Fundamental Research, Mumbai, India

Abstract

We consider a matching problem in a bipartite graph $G = (A \cup B, E)$ where vertices have strict preferences over their neighbors. A matching M is popular if for any matching N, the number of vertices that prefer M is at least the number that prefer N; thus M does not lose a head-to-head election against any matching where vertices are voters. It is easy to find popular matchings; however when there are edge costs, it is NP-hard to find (or even approximate) a min-cost popular matching. This hardness motivates relaxations of popularity.

Here we introduce *fairly popular* matchings. A fairly popular matching may lose elections but there is no good matching (wrt popularity) that defeats a fairly popular matching. In particular, any matching that defeats a fairly popular matching does not occur in the support of any popular mixed matching. We show that a min-cost fairly popular matching can be computed in polynomial time and the fairly popular matching polytope has a compact extended formulation.

We also show the following hardness result: given a matching M, it is NP-complete to decide if there exists a popular matching that defeats M. Interestingly, there exists a set K of at most m popular matchings in G (where $|E| = m$) such that if a matching is defeated by some popular matching in G then it has to be defeated by one of the matchings in K.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Bipartite graphs, Stable matchings, Mixed matchings, Polytopes

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.41

Funding *Telikepalli Kavitha*: Supported by the Department of Atomic Energy, Government of India, under project no. RTI4001.

Acknowledgements Thanks to the reviewers for their helpful comments and suggestions. Thanks to Yuri Faenza and Jaikumar Radhakrishnan for useful discussions.

1 Introduction

Our input is a bipartite graph $G = (A \cup B, E)$ on n vertices and m edges where every vertex has a strict ranking of its neighbors. Such a graph is also called a marriage instance and this is a very well-studied model in two-sided matching markets. A matching M is stable if no edge *blocks* it; edge (a, b) blocks M if both a and b prefer each other to their respective assignments in M. The existence of stable matchings in a marriage instance and the Gale-Shapley algorithm [14] to find one are classical results in algorithms.

Stable matchings are used in many real-world applications such as matching students to schools and colleges [1, 3] and medical residents to hospitals [5, 28]. Stability is a rather strict notion – all stable matchings match the same subset of vertices [15] and the size of a stable matching might be only half the size of a maximum matching. In several applications, the notion of stability can be relaxed to a less demanding notion for the sake of collective welfare.

Popularity is a meaningful relaxation of stability based on empowering *matchings* (instead of edges) to block other matchings. Any pair of matchings, say M and N, can be compared by holding an election between them where every vertex v either casts a vote for the matching in $\{M, N\}$ where it gets a better partner (and being unmatched is its worst choice) or v

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 41; pp. 41:1–41:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

abstains from voting if it is indifferent between M and N. Let $\phi(M,N)$ (resp., $\phi(N,M)$) be the number of votes for M (resp., N). Matching N is *more popular* than matching M (equivalently, N *defeats* M) if $\phi(N,M) > \phi(M,N)$. Let $\Delta(M,N) = \phi(M,N) - \phi(N,M)$.

▶ **Definition 1.** *A matching M is popular if there is no matching more popular than M, i.e., $\Delta(M,N) \geq 0$ for all matchings N in G.*

Gärdenfors [16] introduced the notion of popularity in 1975 where he showed that every stable matching is popular. In fact, stable matchings are min-size popular matchings [18]. Hence relaxing stability to popularity allows larger matchings and more generally, matchings with lower cost (when every edge has a cost) to be feasible.

Several algorithmic and hardness results for popular matchings have been obtained during the last decade and we refer to [6] for a survey. We know efficient algorithms for only a few popular matching problems such as the max-size popular matching problem and the popular edge problem [7, 18, 21]. Many natural optimization problems in popular matchings such as the min-cost popular matching problem are NP-hard [10]; moreover, this problem is NP-hard to approximate to any multiplicative factor. Though relaxing stability to popularity promises matchings with improved optimality, finding these matchings is hard.

The extension complexity of the popular matching polytope of G is $2^{\Omega(m/\log m)}$ [9]. Thus formulating the convex hull of edge incidence vectors of matchings M that satisfy $\Delta(M,N) \geq 0$ for *all* matchings N is hard. This motivates relaxing popularity, i.e., let us waive some constraints $\Delta(M,N) \geq 0$. For what matchings N would it be justified to do so?

Suppose N is "very unpopular" – then N is not a viable alternative and it seems fair to not give N the power to block other matchings. Forbidding *very unpopular* matchings from blocking others is similar in spirit to legal assignments [8] (a relaxation of stable matchings) where only edges that belong to legal assignments are allowed to block matchings. Thus our goal is to come up with a filter that tests matchings for a natural relaxation of popularity and forbid the ones that fail our test to block matchings.

So we seek to identify a subset $\mathcal{S}$ of the set of all matchings in G such that:

(a) Every matching outside $\mathcal{S}$ fails our test that checks for "mild popularity".
(b) It is easy to optimize over matchings M that satisfy $\Delta(M,N) \geq 0$ for all $N \in \mathcal{S}$.
(c) For any matching $T \notin \mathcal{S}$, there is at least one matching $N \in \mathcal{S}$ such that $\Delta(T,N) < 0$.

▶ Remark 2. Note that property (c) is independent of property (a); the latter says every matching $T \notin \mathcal{S}$ has to fail our test of *mild popularity* while the former says any matching $T \notin \mathcal{S}$ has to be defeated by a matching in $\mathcal{S}$, so we will not have $\Delta(T,N) \geq 0$ for all $N \in \mathcal{S}$.

The unpopularity of a matching T is typically measured by its unpopularity factor [27], defined as $u(T) = \max_{N \neq T} \phi(N,T)/\phi(T,N)$. A matching T is popular if and only if $u(T) \leq 1$. Suppose we define a matching T to be very unpopular if $u(T) > k$ for some k. Is it easy to compute a min-cost matching M such that $\Delta(M,N) \geq 0$ for all matchings N with $u(N) \leq k$?

When $k = n-1$, it means that no Pareto optimal matching defeats M – observe that such a matching M has to be popular. So the above problem is NP-hard for $k = n-1$. We show this problem is coNP-hard for $k = 1$ (see Remark 9). Thus using unpopularity factor to come up with a test of mild popularity does not look very promising for tractability.

Our main result. Rather than unpopularity factor, we will use popular *mixed* matchings [26] to define a natural relaxation of popularity. A mixed matching Π is a probability distribution or a lottery over matchings, so $\Pi = \{(M_0,p_0),\ldots,(M_k,p_k)\}$ where $M_0,\ldots,M_k$ are matchings, $p_i > 0$ for all i, and $\sum_{i=0}^{k} p_i = 1$. The notion of popularity can be extended to mixed matchings; the mixed matching Π is popular if $\Delta(\Pi,N) = \sum_{i=0}^{k} p_i \cdot \Delta(M_i,N) \geq 0$ for all matchings N.

The matchings $M_0, \ldots, M_k$ are said to be in the support of $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$. Let us call a matching M *supporting* if there exists a popular mixed matching Π whose support contains M. So every supporting matching participates in some popular lottery over matchings, thus the "supporting" property is a natural relaxation of popularity – we will use this property as our condition for mild popularity. We define *fairly popular* matchings now.

▶ **Definition 3.** *A matching M is fairly popular if $\Delta(M, N) \geq 0$ $\forall$ supporting matchings N.*

For any matching T that defeats a fairly popular matching M, it is the case that even with the help of other matchings, T cannot form a popular mixture. Thus it is natural to regard a *non-supporting* matching T as being "very unpopular". So we set the supporting property as our threshold for mild popularity – thus elections against non-supporting matchings will not be relevant. In other words, even if $\Delta(M, T) < 0$ for a non-supporting matching T, the matching M will continue to be feasible. Intriguingly, waiving the constraints $\Delta(M, T) \geq 0$ for non-supporting matchings T makes the resulting polytope easy to describe.

▶ **Theorem 4.** *Given a marriage instance $G = (A \cup B, E)$ with edge costs, a min-cost fairly popular matching can be computed in polynomial time. Furthermore, the convex hull of edge incidence vectors of fairly popular matchings has a compact extended formulation.*

Key to the above theorem is our characterization of supporting matchings (see Theorem 5). Any point $x \in \mathbb{R}^m_{\geq 0}$ such that $\sum_{e \in \delta(v)} x_e \leq 1$ for each vertex v is a *fractional* matching and x is equivalent to a mixed matching (Birkhoff-von Neumann theorem). A fractional matching x is popular if Π is a popular mixed matching, where Π is any mixed matching that corresponds to x (see [26]). An edge e is a *popular fractional* edge if there exists a popular fractional matching x with $x_e > 0$. Let $E_p \subseteq E$ be the set of popular fractional edges.

Let us call a vertex v *stable* if v is matched in any (equivalently, every [15]) stable matching in G. So unstable vertices are those left unmatched in every stable matching.

▶ **Theorem 5.** *Let $G = (A \cup B, E)$ be a marriage instance and let M be a matching in G. The following three statements are equivalent.*

1. *M is supporting, i.e., M occurs in the support of some popular mixed matching.*
2. *No popular mixed matching defeats M, i.e., $\Delta(\Pi, M) = 0$ $\forall$ popular mixed matchings Π.*
3. *M matches all stable vertices and $M \subseteq E_p$.*

▶ Remark 6. Theorem 5 implies that any matching that is *non-supporting* is defeated by some popular mixed matching and thus, by some supporting matching (since every popular mixed matching is a lottery over supporting matchings). So $\mathcal{S}$ = {supporting matchings} satisfies properties (a), (b), and (c) stated earlier. Thus every fairly popular matching is also supporting.

Observe that the set of popular matchings does not satisfy the property that any matching outside this set has to be defeated by at least one matching in this set. That is, it is not the case that every *unpopular* matching has to lose to one or more popular matchings. For example, consider the following instance where $A = \{a_0, a_1, a_2\}$ and $B = \{b_0, b_1\}$.

$a_0\colon\ b_0 \succ b_1$	$a_1\colon\ b_0 \succ b_1$	$a_2\colon\ b_1$
$b_0\colon\ a_0 \succ a_1$	$b_1\colon\ a_0 \succ a_1 \succ a_2$	

Here a_0 and b_0 are each other's top choice neighbors and a_0's second choice is b_1 and b_0's second choice is a_1 and so on. The above instance has only one popular matching $P = \{(a_0, b_0), (a_1, b_1)\}$. The matching $M = \{(a_0, b_1), (a_1, b_0)\}$ is not popular since the matching $N = \{(a_0, b_0), (a_2, b_1)\}$ is more popular than M; the vertices a_0, b_0, a_2 prefer N while a_1, b_1 prefer M. Observe that the popular matching P is *not* more popular than M.

Interestingly, M is a supporting matching since the mixed matching $\Pi = \{(M, \frac{1}{2}), (P, \frac{1}{2})\}$ is popular. Moreover, M is fairly popular since N is the only matching that defeats M and observe that N leaves the stable vertex a_1 unmatched, hence N is *not* a supporting matching.

A hardness result. As observed above, it is not the case that every unpopular matching has to be defeated by some popular matching. This motivates the following question: how easy is it to decide if there exists a popular matching that defeats a given matching M? This is a natural question when matching M is already in place and we want to replace M with a popular matching. An ideal matching would be a popular matching that is more popular than M, if such a matching exists. Interestingly, we can show a "compactness" result. Note that G may have more than 2^n popular matchings [32].

▶ **Proposition 7.** *There is a set K of at most m popular matchings in G such that any matching defeated by some popular matching in G has to be defeated by a matching in K.*

However, deciding if there is a popular matching that defeats a given matching is hard.

▶ **Theorem 8.** *Given a marriage instance $G = (A \cup B, E)$ and a matching M in G, it is NP-complete to decide if there exists any popular matching that is more popular than M.*

▶ Remark 9. It was mentioned earlier that it is coNP-hard to compute a min-cost matching that is not defeated by any popular matching. This hardness follows from Theorem 8 by setting $\mathsf{cost}(e) = 0$ for each $e \in M$ and $\mathsf{cost}(e) = 1$ for any $e \notin M$.

For any matching M, if there is a popular matching that defeats M then it is natural to regard M as a very unpopular matching (as there is a popular matching better than M). However to define a *mildly popular* matching as one that is undefeated by popular matchings would not have been very helpful as we know it is coNP-hard to identify such matchings (by Theorem 8). A natural strengthening of this property would have been to say that a matching M is mildly popular if and only if M is undefeated by popular *mixed* matchings. This is precisely one of the characterizations of supporting matchings (by Theorem 5).

Related results. The min-cost stable matching problem is very well-studied with several polynomial time algorithms [11, 12, 13, 20, 33] to solve this problem; furthermore, the stable matching polytope has a simple and elegant linear size formulation in $\mathbb{R}^m$ [29, 31]. It is known that the popular fractional matching polytope of G is half-integral [19].

A min-cost popular matching in G can be computed in $O^*(2^{n/4})$ time [25]. The intractability of the min-cost popular matching problem has motivated relaxations such as *quasi-popularity* [9] and *semi-popularity* [25]. A matching M is quasi-popular if $u(M) \leq 2$. Computing a min-cost quasi-popular matching is NP-hard; however a quasi-popular matching of cost at most that of a min-cost popular matching can be computed in polynomial time [9]. A matching M is semi-popular if $\Delta(M, N) \geq 0$ for at least half the matchings N in G. A bicriteria approximation algorithm was given in [25] to find an *almost* semi-popular matching whose cost is at most twice the cost of a min-cost popular matching.

Our techniques. The characterization of supporting matchings (given in Section 2) uses the half-integrality of the popular fractional matching polytope in a marriage instance [19] along with Hall's theorem. A technical lemma used here (and proved in the appendix) is based on the existence of certain helpful stable matchings as shown in [17].

Our characterization of supporting matchings implies that a matching M is fairly popular if and only if $M = \cup_C M_c$, where C is a connected component in the subgraph $(A \cup B, E_p)$ and every matching M_c in this decomposition has a certain *witness* or dual certificate. We show a surjective mapping from the union of sets of stable matchings in two auxiliary graphs G'_c and G''_c to the set of such matchings M_c. Let $\mathcal{S}'_c$ (resp., $\mathcal{S}''_c$) be the stable matching polytope of G'_c (resp., G''_c). The convex hull of $\mathcal{S}'_c \cup \mathcal{S}''_c$ is an extension of the convex hull of edge incidence vectors of such matchings M_c. Using Balas' theorem [2] to formulate the convex hull of $\mathcal{S}'_c \cup \mathcal{S}''_c$ leads to Theorem 4 proved in Section 3.

The LP-machinery for popular matchings was introduced in [26] and used in [19, 22] to study popular fractional matchings. The graphs G'_c and G''_c are inspired by instances from [7, 23, 24] that solve variants of the popular matching problem by modeling them as stable matching problems in appropriate graphs. Our novelty is in our characterization of supporting matchings – this leads to a characterization of fairly popular matchings which allows us to formulate an extension of the fairly popular matching polytope $\mathcal{F}$ with poly(m,n) many constraints, i.e., we show the polytope $\mathcal{F}$ has a compact extended formulation.

Our NP-hardness proof (given in Section 4) is based on the NP-hardness (from [10]) of deciding if there exists a popular matching that contains a given pair of edges.

2 A Characterization of Supporting Matchings

We prove Theorem 5 in this section. Before we characterize supporting matchings, it will be useful to recall some properties of popular fractional matchings in a marriage instance G.

A fractional matching x in G is a convex combination of matchings (by Birkhoff-von Neumann theorem). Recall that x is popular if Π is a popular mixed matching, where Π is any mixed matching that is equivalent to x. Alternatively, as shown in [26], x is popular if $\Delta(x, M) \ge 0$ for all matchings M where $\Delta(x, M) = \sum_{u \in A \cup B} \mathsf{vote}_u(x, M)$ and $\mathsf{vote}_u(x, M)$ is u's fractional vote (a value in $[-1, 1]$) for its assignment in x versus its assignment in M. Section 4 has more details on comparing a matching M with a fractional matching x.

The popular fractional matching polytope of G is the convex hull of all popular fractional matchings in G. It was shown in [19] that the popular fractional matching polytope of G is half-integral. The proof of half-integrality uses the graph $H = (A_H \cup B_H, E_H)$ defined below.

The graph H can be regarded as consisting of *two* copies of $G = (A \cup B, E)$ (see Figure 1). The vertex set $A_H = A_0 \cup B_1$ and $B_H = B_0 \cup A_1$, where $A_i = \{a_i : a \in A\}$ and $B_i = \{b_i : b \in B\}$ for $i = 0, 1$. The edge set E_H of H is described below.

- For every $(a, b) \in E$, there are 2 edges (a_0, b_0) and (a_1, b_1) in E_H.
- For every $u \in A \cup B$, there is a single edge (u_0, u_1) in E_H.

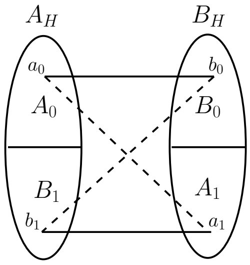


Figure 1 The vertex set of H has 2 copies u_0 and u_1 of every vertex u in G.

For any $u \in A \cup B$: if u's preference order in G is $v \succ v' \succ \cdots \succ v''$ then u_i's preference order (for $i = 0, 1$) in H is $v_i \succ v'_i \succ \cdots \succ v''_i \succ u_{1-i}$; so u_i's last choice neighbor is u_{1-i}.

The graph H admits a *perfect* stable matching, i.e., one that matches all vertices. Let S be any stable matching in G. Consider the matching S' in H defined as $S_0 \cup S_1 \cup \{(u_0, u_1) : u$ is unmatched in $S\}$ where $S_i = \{(a_i, b_i) : (a, b) \in S\}$ for $i = 0, 1$. It is easy to see that S' is a perfect stable matching in H.

It was shown in [19, Theorem 2] that if a marriage instance has a perfect stable matching then its popular fractional matching polytope is integral. Thus the popular fractional matching polytope of H is integral.

The function f. For any matching N in G, there is a corresponding matching N' in H defined as $\{(a_0, b_0), (a_1, b_1) : (a, b) \in N\} \cup \{(u_0, u_1) : u \text{ is unmatched in } N\}$. This map extends to fractional matchings, so for any fractional matching x in G, there is a corresponding fractional matching x' in H. Similarly, there is a map f from the set of fractional matchings in H to the set of fractional matchings in G: $f(y) = x$ where $x_{(a,b)} = (y_{(a_0,b_0)} + y_{(a_1,b_1)})/2$ for any $(a, b) \in E$. Observe that $f(x') = x$ where x' is the fractional matching in H that corresponds to x in G. If the fractional matching y is popular in H then the fractional matching $f(y)$ is popular in G since $\Delta(f(y), N) = \Delta(y, N')/2$ for any matching N in G.

Note that $(a, b) \in E$ is a popular fractional edge in G, i.e., $(a, b) \in E_p$, if and only if (a_0, b_0) and (a_1, b_1) are popular fractional edges in H. Since the popular fractional matching polytope of H is integral, it follows that (a_0, b_0) and (a_1, b_1) are *popular edges*[1] in H. Also, (u_0, u_1) is a popular edge in H if and only if u is an unstable vertex in G.

Proof of Theorem 5

We need to show the following three statements are equivalent.

1. M is supporting.
2. No popular mixed matching defeats M.
3. M matches all stable vertices and $M \subseteq E_p$.

Proof of 1⇒2. Let M be a supporting matching. Then there exists a popular mixed matching $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$ where $M = M_i$ for some i. Suppose there is a popular mixed matching Π' that defeats M, i.e., $\Delta(\Pi', M) > 0$. Because both Π and Π' are popular mixed matchings, we have $\Delta(\Pi', \Pi) = \sum_j p_j \cdot \Delta(\Pi', M_j) = 0$. Since $\Delta(\Pi', M_i) > 0$ and $\Delta(\Pi', \Pi) = 0$, there has to exist some matching M_j on which Π has support such that $\Delta(\Pi', M_j) < 0$. However this contradicts Π''s popularity, thus 1⇒2.

Proof of 2⇒3. This part needs the following technical lemma. The proof of Lemma 10 uses the existence of certain helpful stable matchings as shown in [17] and is given in the appendix. Call an edge e *unpopular* if there exists no popular matching that contains e.

▶ **Lemma 10.** *Any matching in H that contains an unpopular edge is defeated by some popular matching in H.*

Let M be a matching in G such that either M has an edge not in E_p or some stable vertex is left unmatched in M. So the matching $M' = \{(a_0, b_0), (a_1, b_1) : (a, b) \in M\} \cup \{(u_0, u_1) : u$ is unmatched in $M\}$ in H has an edge that is not a popular edge. Then some popular matching P in H defeats M' (by Lemma 10).

Recall the map f from the set of fractional matchings in H to the set of fractional matchings in G defined earlier in Section 2. Let $r = f(P)$. The fractional matching r is popular in G because P is a popular matching in H. Since $\Delta(P, M') > 0$, we have $\Delta(r, M) > 0$. The fractional matching r can be regarded as a mixed matching Π; moreover, Π is popular since r is popular. Thus there is a popular mixed matching Π that is more popular than M, a contradiction to M satisfying property 2. Thus 2⇒3.

[1] An edge e is popular if there is a popular matching that contains e.

Proof of 3⇒1. Let $e = (a, b) \in M$. Since $M \subseteq E_p$, by what was discussed earlier in Section 2, there are popular matchings M^0_e and M^1_e in H that contain (a_0, b_0) and (a_1, b_1), respectively. For any vertex u left unmatched in M, it has to be the case that u is an unstable vertex in G. So there is a popular matching M_u in H that contains (u_0, u_1).

Suppose $M = \{e_1, \ldots, e_\ell\}$ and let $u_1, \ldots, u_t$ be left unmatched in M. Consider the $2\ell + t$ matchings $M^0_{e_1}, \ldots, M^0_{e_\ell}, M^1_{e_1}, \ldots, M^1_{e_\ell}$ and $M_{u_1}, \ldots, M_{u_t}$ in H analogous to the matchings M^0_e, M^1_e, and M_u defined above. Let H' be the graph whose edge set is the multiset of edges present in these $2\ell + t$ matchings, i.e., multiple copies of an edge are present in this edge set if this edge is present in more than one matching. The graph H' is $(2\ell + t)$-regular since each of these $2\ell + t$ matchings is popular and hence, perfect in H (recall that H has a perfect stable matching and stable matchings are min-size popular matchings).

Observe that $M' = \{(a_0, b_0), (a_1, b_1) : (a, b) \in M\} \cup \{(u_0, u_1) : u \text{ is unmatched in } M\}$ belongs to H'. Delete M' from H'. Since M' is a perfect matching in H', the resulting graph $H'' = H' \setminus M'$ is $(2\ell + t - 1)$-regular. It follows from Hall's theorem that H'' can be decomposed into $2\ell + t - 1$ perfect matchings $N'_1, \ldots, N'_{2\ell+t-1}$. Thus we have:

$$I_{M'} + I_{N'_1} + \cdots + I_{N'_{2\ell+t-1}} = I_{M^0_{e_1}} + \cdots + I_{M^1_{e_\ell}} + I_{M_{u_1}} + \cdots + I_{M_{u_t}},$$

where for any matching N, the vector I_N is its edge incidence vector.

The $2\ell + t$ matchings $M^0_{e_1}, \ldots, M^1_{e_\ell}, M_{u_1}, \ldots, M_{u_t}$ (on the right hand side above) are popular in H. Hence the fractional matching $q = (I_{M^0_{e_1}} + \cdots + I_{M_{u_t}})/(2\ell + t)$, which can also be written as $(I_{M'} + I_{N'_1} + \cdots + I_{N'_{2\ell+t-1}})/(2\ell + t)$, is popular in H.

So $r = f(q)$ is a popular fractional matching in G. The mixed matching $\Pi = \{(M, \frac{1}{2\ell+t}), \ldots\}$ is equivalent to r and it has support on M. Moreover, Π is a popular mixed matching since r is a popular fractional matching. Thus M is a supporting matching. Hence 3⇒1. ◀

3 The Fairly Popular Matching Polytope

We prove Theorem 4 in this section. We will see an LP framework for fairly popular matchings in Section 3.1. A characterization of fairly popular matchings will be given in Section 3.2. In Sections 3.3 and 3.4, this characterization will be used to solve the min-cost fairly popular matching problem in polynomial time.

3.1 An LP Framework

Our input instance is $G = (A \cup B, E)$. Let $E_p \subseteq E$ be the set of popular fractional edges in G. The set E_p can be computed in linear time by running the popular edge algorithm (from [7]) in the instance H described in Section 2.

Let $\tilde{E}_p = E_p \cup \{(u, u) : u \text{ is an unstable vertex in } G\}$ and let $G_p = (A \cup B, \tilde{E}_p)$. We know from Theorem 5 that every perfect matching $\tilde{N}$ in G_p is a supporting matching N augmented with self-loops at vertices left unmatched in N; conversely, every supporting matching N augmented with self-loops at unmatched vertices is a perfect matching $\tilde{N}$ in G_p.

Let M be any matching in G. In order to decide if there exists a supporting matching that defeats M, the following edge weight function in G_p will be useful. For any $(a, b) \in E_p$:

$$\text{let } \mathsf{wt}_M(a, b) = \begin{cases} 2 & \text{if } (a, b) \text{ is a blocking edge to } M; \\ -2 & \text{if } a \text{ and } b \text{ prefer their partners in } M \text{ to each other;} \\ 0 & \text{otherwise.} \end{cases}$$

For any unstable vertex u, let $\mathsf{wt}_M(u, u) = 0$ if u is left unmatched in M, else $\mathsf{wt}_M(u, u) = -1$.

Consider the following linear program (LP1). For any vertex v, let $\delta_p(v)$ be the set of edges incident to v in G_p.

$$\text{maximize} \sum_{e \in \tilde{E}_p} \mathsf{wt}_M(e) \cdot x_e \tag{LP1}$$

$$\text{subject to}$$

$$\sum_{e \in \delta_p(v)} x_e = 1 \quad \forall\, v \in A \cup B \quad \text{and} \quad x_e \geq 0 \quad \forall\, e \in \tilde{E}_p.$$

Since the constraint matrix is totally unimodular, (LP1) is integral. This LP computes a max-weight perfect matching $\tilde{N}$ in G_p (so N is supporting by Theorem 5) with respect to the edge weight function wt_M. The following claim is easy to see.

▷ Claim 11. For any perfect matching $\tilde{N}$ in G_p, we have $\mathsf{wt}_M(\tilde{N}) = \Delta(N, M)$.

Proof. For any edge $e = (a, b) \in E_p$, observe that $\mathsf{wt}_M(e) = \mathsf{vote}_a(b, M) + \mathsf{vote}_b(a, M)$ where for any vertex v and neighbor v', $\mathsf{vote}_v(v', M) \in \{\pm 1, 0\}$ is v's vote for v' versus its assignment in M. So $\mathsf{vote}_v(v', M) = 1$ if v prefers v' to its assignment in M, it is -1 if v prefers its assignment in M to v', otherwise it is 0. Similarly, for any unstable vertex v, $\mathsf{wt}_M(v, v)$ is 0 if M leaves v unmatched, else it is -1.

Hence for any perfect matching $\tilde{N}$ in G_p, observe that $\mathsf{wt}_M(\tilde{N})$ is the sum of votes of all vertices, where each vertex votes for its assignment in N versus its assignment in M. In other words, $\mathsf{wt}_M(\tilde{N}) = \phi(N, M) - \phi(M, N) = \Delta(N, M)$. ◁

It follows from Claim 11 that if the optimal value of (LP1) is positive then there exists a supporting matching that defeats M; else $\Delta(N, M) \leq 0$ for all supporting matchings N, so M is fairly popular. Note that for any stable matching N in G, we have $\mathsf{wt}_M(\tilde{N}) = \Delta(N, M) \geq 0$ (due to N's popularity in G). So the optimal value of (LP1) has to be at least 0. Hence M is fairly popular if and only if the optimal value of (LP1) is 0.

Let $U \subseteq A \cup B$ be the set of unstable vertices in G. The linear program (LP2) is the dual LP.

$$\text{minimize} \sum_{v \in A \cup B} \alpha_v \tag{LP2}$$

$$\text{subject to}$$

$$\alpha_a + \alpha_b \;\geq\; \mathsf{wt}_M(a, b) \;\; \forall\, (a, b) \in E_p \qquad \text{and} \qquad \alpha_u \;\geq\; \mathsf{wt}_M(u, u) \;\; \forall\, u \in U.$$

So M is fairly popular if and only if the optimal value of (LP2) is 0.

3.2 Witnesses for Fairly Popular Matchings

Let C be any connected component in $G_p = (A \cup B, \tilde{E}_p)$. Since all stable matchings in G match the stable vertices of C among themselves, the number of stable vertices in $C_A = C \cap A$ is the same as the number of stable vertices in $C_B = C \cap B$. Hence there are k stable vertices in C_A if and only if there are k stable vertices in C_B.

► **Lemma 12.** *A matching M is fairly popular if and only if there exists a feasible solution α to* (LP2) *such that for every connected component C in G_p, we have $\sum_{v \in C} \alpha_v = 0$ and furthermore,*

- *either $\alpha_v \in \{0, \pm 2, \pm 4, \ldots, \pm 2k\}$ for all $v \in C$*
- *or $\alpha_v \in \{\pm 1, \pm 3, \pm 5, \ldots, \pm(2k+1)\}$ for all $v \in C$,*

where $2k$ is the number of stable vertices in C.

Proof. Let M be a matching such that there exists a feasible solution α to (LP2) with $\sum_{v \in C} \alpha_v = 0$ for every connected component C in G_p. Then $\sum_{v \in A \cup B} \alpha_v = 0$ and so the optimal value of (LP2) is 0. Hence M is fairly popular.

Conversely, let M be a fairly popular matching in G and let α be an optimal solution to (LP2). The constraint matrix of (LP2) is totally unimodular, so we can assume that $\alpha \in \mathbb{Z}^n$.

Let C be any connected component in G_p. We have $\mathsf{wt}_M(\tilde{N}_c) \ge 0$ where N is any stable matching in G and $N_c = N \cap (C \times C)$. Hence $\sum_{v \in C} \alpha_v \ge 0$. Moreover, $\sum_C \sum_{v \in C} \alpha_v = \sum_{v \in A \cup B} \alpha_v = 0$ since M is fairly popular. Hence it has to be the case that $\sum_{v \in C} \alpha_v = 0$ for every connected component C in G_p.

Every edge in E_p belongs to some popular fractional matching in G. Let q be the popular fractional matching that $(a, b) \in E_p$ belongs to, where a and b are vertices in C. We have $\Delta(q, M) = 0$ since q is a popular fractional matching, thus q is an optimal solution to (LP1). Because α is an optimal solution to (LP2), we have $\alpha_a + \alpha_b = \mathsf{wt}_M(a, b)$ by complementary slackness, i.e., every edge in G_p is tight. So $\alpha_a + \alpha_b = \mathsf{wt}_M(a, b) \in \{0, \pm 2\}$ for all $(a, b) \in E_p$. Hence the α-values of all the vertices in C have the same parity.

Suppose every vertex of C is stable. Then we can update the α-values of vertices in C as follows for any value t: $\alpha_a = \alpha_a - t$ for all $a \in C_A$ and $\alpha_b = \alpha_b + t$ for all $b \in C_B$. The updated α-values are also a feasible solution to (LP2) since the sum $\alpha_a + \alpha_b$ for any $(a, b) \in E_p$ (where a and b are in C) is unchanged by this update; moreover, we assumed that C has no unstable vertex, so there is no constraint $\alpha_u \ge \mathsf{wt}_M(u, u)$ for any $u \in C$.

Moreover, the sum of α-values of all vertices in C is unchanged by this update since $|C_A| = |C_B| = k$ (because C has only stable vertices), so $\sum_{v \in C} \alpha_v = 0$. Thus we can preserve optimality and shift α-values so as to make $\alpha_v = 0$ for some $v \in C$. All the edges in G_p are tight, so the matched partners of vertices with α-value 0 also have α-value 0 and all neighbors in C of vertices with α-value 0 have their α-values in $\{0, \pm 2\}$. Their partners have α-values in $\{0, \pm 2\}$ and neighbors of these vertices have α-values in $\{0, \pm 2, \pm 4\}$ and so on. Since the number of stable vertices in C_A (and also in C_B) is k, we can conclude that there exists an optimal solution α to (LP2) such that $\alpha_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$.

Let us now assume that C has at least one unstable vertex. Consider the matching $\tilde{N} = N \cup \{(u, u) : u \in U\}$, where N is any stable matching in G and U is the set of unstable vertices in G. The matching $\tilde{N}$ is an optimal solution to (LP1). By complementary slackness, we have $\alpha_u = \mathsf{wt}_M(u, u)$ for every $u \in U$. Hence $\alpha_u \in \{0, -1\}$ for every $u \in U$. Since the α-values of all the vertices in C have the same parity, we have the following two cases.

Case 1. The α-values of all the vertices in C are even. Then $\alpha_u = 0$ for every $u \in U \cap C$. As argued above (when C had no unstable vertex), this implies that $\alpha_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$.

Case 2: The α-values of all the vertices in C are odd. Then $\alpha_u = -1$ for every $u \in U \cap C$. An analogous argument to the one above shows that $\alpha_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\}$ for all $v \in C$. ◀

A characterization of fairly popular matchings. By Lemma 12, a matching M is fairly popular if and only if $M = \cup_C M_c$ where for every connected component C in G_p, there exists γ (this is the vector α in Lemma 12 restricted to vertices in C) such that:

1. $\sum_{v \in C} \gamma_v = 0$;
2. $\gamma_a + \gamma_b \ge \mathsf{wt}_{M_c}(a, b)$ for $(a, b) \in E_p \cap (C \times C)$ and $\gamma_u \ge \mathsf{wt}_{M_c}(u, u)$ for $u \in U \cap C$;
3. either $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$ or $\gamma_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\}$ for all $v \in C$, where $2k$ is the number of stable vertices in C.

Witnesses. We know that M is fairly popular if and only if for each connected component C in G_p, there exists γ such that $M_c = M \cap (C \times C)$ and γ satisfy properties 1-3 given above. Such a vector γ will be called a *witness* of M_c. Let $G_c = (C, E_c)$ where $E_c = E_p \cap (C \times C)$.

▶ **Definition 13.** *Call a matching M_c in G_c* valid *if it has a witness, i.e., there exists a vector γ such that M_c and γ satisfy properties 1-3 given above.*

Let $\mathcal{F}_c$ be the convex hull of edge incidence vectors of all valid matchings in G_c. By Lemma 12, $\mathcal{F}_c$ is the convex hull of $\mathcal{F}_c^0 \cup \mathcal{F}_c^1$ where:

- $\mathcal{F}_c^0$ is the convex hull of edge incidence vectors of valid matchings in G_c with a witness γ such that $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$.
- $\mathcal{F}_c^1$ is the convex hull of edge incidence vectors of valid matchings in G_c with a witness γ such that $\gamma_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\}$ for all $v \in C$.

3.3 Two Useful Stable Matching Instances

Let C be any connected component in G_p with $|C| \geq 2$. We will now describe instances G'_c and G''_c such that the stable matching polytope of G'_c (resp., G''_c) is an extension of $\mathcal{F}_c^0$ (resp., $\mathcal{F}_c^1$). Let S be the set of stable vertices in G and let $|S \cap C| = 2k$.

The instance $G'_c = (A'_c \cup B'_c, E'_c)$. Every $a \in S \cap C_A$ has $2k+1$ copies $a_{-k}, \ldots, a_0, \ldots, a_k$ in A'_c. Recall that U is the set of unstable vertices in G. Every $a \in U \cap C_A$ has exactly one copy a_0 in A'_c.

Let $B'_c = \{\tilde{b} : b \in C_B\} \cup \{d_{1-k}(a), \ldots, d_k(a) : a \in S \cap C_A\}$, where the set $\{\tilde{b} : b \in C_B\}$ is a copy of C_B. Along with vertices in $\{\tilde{b} : b \in C_B\}$, the set B'_c contains $2k$ *dummy* vertices $d_{1-k}(a), \ldots, d_k(a)$ for each $a \in S \cap C_A$. The purpose of the $2k$ dummy vertices $d_{1-k}(a), \ldots, d_k(a)$ is to ensure that only one of $a_{-k}, \ldots, a_{-1}, a_0, a_1, \ldots, a_k$ is matched to a *non-dummy* neighbor in any stable matching in G'_c.

For any $a \in S \cap C_A$, the set E'_c has the edges $(a_{i-1}, d_i(a))$ and $(a_i, d_i(a))$ for $1-k \leq i \leq k$. For every edge (a, b) in E_c, the following edges are in E'_c. Since vertices in U form an independent set, note that at least one of a, b has to be in S.

1. If only one of a, b is in S then there is only one edge $(a_0, \tilde{b})$ in E'_c.
2. If both a and b are in S then there are $2k+1$ edges $(a_i, \tilde{b})$ in E'_c where $-k \leq i \leq k$.

Let a's preference order among its neighbors in G_c be $b_1 \succ \cdots \succ b_r$.

- If $a \in U$ then the preference order of a_0 is $\tilde{b}_1 \succ \cdots \succ \tilde{b}_r$.
- Suppose $a \in S$. The vertex a_0's preference order is $d_0(a) \succ \tilde{b}_1 \succ \cdots \succ \tilde{b}_r \succ d_1(a)$. Note that all of a's neighbors in G_c are present in a_0's preference list – this will not be so for a_i, where $i \neq 0$. Let $t_1, \ldots, t_s$ be a's neighbors in G_c that are in S. Let a's preference order among these neighbors be $t_1 \succ \cdots \succ t_s$.
  - a_{-k}'s preference order in G'_c is $\tilde{t}_1 \succ \cdots \succ \tilde{t}_s \succ d_{1-k}(a)$.
  - For $i \in \{1-k, \ldots, k-1\} \setminus \{0\}$: a_i's preference order is $d_i(a) \succ \tilde{t}_1 \succ \cdots \succ \tilde{t}_s \succ d_{i+1}(a)$.
  - a_k's preference order in G'_c is $d_k(a) \succ \tilde{t}_1 \succ \cdots \succ \tilde{t}_s$.

  For any i, the preference order of $d_i(a)$ is $a_{i-1} \succ a_i$.

Consider any $b \in C_B$. Let b's preference order for its neighbors in G_c be $a \succ \cdots \succ z$. If $b \in U$ then $\tilde{b}$'s preference order for its neighbors in G'_c is $a_0 \succ \cdots \succ z_0$.

Suppose $b \in S$. Let $\{a', \ldots, z'\} \subseteq \{a, \ldots, z\}$ be the set of b's neighbors in G_c that are in S. Let b's preference order among these neighbors be $a' \succ \cdots \succ z'$. The preference order of $\tilde{b}$ in G'_c is:

$$\underbrace{a'_k \succ \cdots \succ z'_k}_{\text{level } k \text{ neighbors}} \succ \cdots \succ \underbrace{a'_1 \succ \cdots \succ z'_1}_{\text{level 1 neighbors}} \succ \underbrace{a_0 \succ \cdots \succ z_0}_{\text{level 0 neighbors}} \succ \cdots \succ \underbrace{a'_{-k} \succ \cdots \succ z'_{-k}}_{\text{level } -k \text{ neighbors}}$$

So copies of all neighbors of b in G_c are present only in level 0. Note that $\tilde{b}$ prefers subscript/level i neighbors to level j neighbors for any $i > j$.

Stable matchings in G'_c. For any valid matching M_c in G_c with a witness γ such that $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$, define M'_c in G'_c as follows. For every $(a, b) \in M_c$:

- include the edge $(a_i, \tilde{b})$ in M'_c where $\gamma_a = -2i$;
- for $j < i$ and $a \in S$ do: add the edge $(a_j, d_{j+1}(a))$ to M'_c;
- for $j > i$ and $a \in S$ do: add the edge $(a_j, d_j(a))$ to M'_c.

We will show in Lemma 14 that M'_c is a stable matching in G'_c. Conversely, let M'_c be any stable matching in G'_c. Let M_c be the *preimage* of M'_c, i.e., M_c is obtained by deleting all edges in M'_c that are incident to dummy vertices and replacing any edge $(a_i, \tilde{b}) \in E'_c$ with $(a, b) \in E_c$. Note that M_c is a matching in G_c because all dummy vertices (being top choice neighbors) have to be matched in any stable matching in G'_c and so at most one of the a_i's can be matched to a non-dummy neighbor in M'_c.

We will show in Lemma 14 that M_c is a valid matching in G_c. The proof of Lemma 14 uses ideas from [23, 24] and is given in the appendix.

▶ **Lemma 14.** *M_c is a valid matching in G_c with a witness γ such that $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$ if and only if M'_c is a stable matching in G'_c.*

The instance $G''_c = (A''_c \cup B''_c, E''_c)$. Every $a \in S \cap C_A$ has $2k + 2$ copies $a_{-k}, \ldots, a_{-1}, a_0, \ldots, a_{k+1}$ in A''_c. Every $a \in U \cap C_A$ has $k + 2$ copies $a_{-k}, \ldots, a_{-1}, a_0, a_1$ in A''_c. Let $B''_c = \{\tilde{b} : b \in C_B\} \cup \{d_{1-k}(a), \ldots, d_{k+1}(a) : a \in S \cap C_A\} \cup \{d_{1-k}(a), \ldots, d_1(a) : a \in U \cap C_A\}$. As before, the set $\{\tilde{b} : b \in C_B\}$ is a copy of the set C_B. Along with vertices in $\{\tilde{b} : b \in C_B\}$, the set B''_c contains $2k + 1$ dummy vertices for each $a \in S \cap C_A$ and $k + 1$ dummy vertices for each $a \in U \cap C_A$. For each edge $(a, b) \in E_c$, the following edges are in E''_c:

1. If $a \in U$ (so $b \in S$) then there are $k + 2$ edges $(a_i, \tilde{b})$ in E''_c where $-k \le i \le 1$.
2. If $b \in U$ (so $a \in S$) then there are $k + 2$ edges $(a_i, \tilde{b})$ in E''_c where $0 \le i \le k + 1$.
3. If both a and b are in S then there are $2k + 2$ edges $(a_i, \tilde{b})$ in E''_c where $-k \le i \le k + 1$.

Let $a \in C_A$. The set E''_c also has the edges $(a_{i-1}, d_i(a))$ and $(a_i, d_i(a))$ for $1-k \le i \le k+1$ if $a \in S$ and for $1 - k \le i \le 1$ if $a \in U$. For any i, the preference order of $d_i(a)$ is $a_{i-1} \succ a_i$.

Let a's preference order among its neighbors in G_c be $b_1 \succ \cdots \succ b_r$. Let $t_1, \ldots, t_s$ be a's neighbors in G_c that are in S and let $t_1 \succ \cdots \succ t_s$ be a's preference order among these neighbors.

- a_{-k}'s preference order in G''_c is $\tilde{t}_1 \succ \cdots \succ \tilde{t}_s \succ d_{1-k}(a)$.
- a_i's preference order is $d_i(a) \succ \tilde{t}_1 \succ \cdots \succ \tilde{t}_s \succ d_{i+1}(a)$ for $1 - k \le i \le -1$.
- If $a \in U$ then all of a's neighbors are in S and a_0's preference order is $d_0(a) \succ \tilde{b}_1 \succ \cdots \succ \tilde{b}_r \succ d_1(a)$ and a_1's preference order is $d_1(a) \succ \tilde{b}_1 \succ \cdots \succ \tilde{b}_r$.
- If $a \in S$ then a_i's preference order is $d_i(a) \succ \tilde{b}_1 \succ \cdots \succ \tilde{b}_r \succ d_{i+1}(a)$ for $0 \le i \le k$ and a_{k+1}'s preference order is $d_{k+1}(a) \succ \tilde{b}_1 \succ \cdots \succ \tilde{b}_r$.

Consider any $b \in C_B$. Let b's preference order for its neighbors in G_c be $a \succ \cdots \succ z$. If $b \in U$ then $a, \ldots, z$ are in S and $\tilde{b}$'s preference order among its neighbors in G''_c is:

$$\underbrace{a_{k+1} \succ \cdots \succ z_{k+1}}_{\text{level } k+1 \text{ neighbors}} \succ \underbrace{a_k \succ \cdots \succ z_k}_{\text{level } k \text{ neighbors}} \succ \cdots \succ \underbrace{a_1 \succ \cdots \succ z_1}_{\text{level 1 neighbors}} \succ \underbrace{a_0 \succ \cdots \succ z_0}_{\text{level 0 neighbors}}$$

Suppose $b \in S$. Let $a', \ldots, z'$ be b's neighbors in G_c that are in S and let b's preference order among these neighbors be $a' \succ \cdots \succ z'$. Then the preference order of $\tilde{b}$ in G''_c is:

$$\underbrace{a'_{k+1} \succ \cdots \succ z'_{k+1}}_{\text{level } k+1 \text{ neighbors}} \succ \cdots \succ \underbrace{a'_2 \succ \cdots \succ z'_2}_{\text{level 2 neighbors}} \succ \underbrace{a_1 \succ \cdots \succ z_1}_{\text{level 1 neighbors}} \succ \cdots \succ \underbrace{a_{-k} \succ \cdots \succ z_{-k}}_{\text{level } -k \text{ neighbors}}$$

Note that copies of all neighbors of b in G_c are present in level i only for $-k \leq i \leq 1$.

Stable matchings in G''_c. For any valid matching M_c in G_c with a witness γ such that $\gamma_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\}$ for all $v \in C$, define M''_c in G''_c as follows. For every $(a, b) \in M_c$:

- include the edge $(a_i, \tilde{b})$ in M''_c where $\gamma_a = -(2i - 1)$;
- for $j < i$ do: add the edge $(a_j, d_{j+1}(a))$ to M''_c;
- for $j > i$ do: add the edge $(a_j, d_j(a))$ to M''_c.

We will show that M''_c is a stable matching in G''_c. Conversely, let M''_c be any stable matching in G''_c. As before, let M_c be the preimage of M''_c; observe that M_c is a matching in G_c. Lemma 15 (proved in the appendix) shows that M_c is a valid matching in G_c.

▶ **Lemma 15.** *M_c is a valid matching in G_c with a witness γ such that $\gamma_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\}$ for all $v \in C$ if and only if M''_c is a stable matching in G''_c.*

3.4 A compact extended formulation

For any vertex v in G'_c, let $\delta'_c(v)$ be the set of edges incident to v in G'_c and for any neighbor u of v, let $\{w \succ_v u\}$ be the set of all neighbors of v in G'_c that v prefers to u. Let T'_c be the set of vertices in G'_c matched in any stable matching in this graph. Consider constraints (1)-(3) in variables y_e where $e \in E'_c$ and λ_c (this variable will be defined later).

$$\sum_{w\colon w \succ_{a_i} \tilde{b}} y_{(a_i, w)} + \sum_{s\colon s \succ_{\tilde{b}} a_i} y_{(s, \tilde{b})} + y_{(a_i, \tilde{b})} \quad \geq \quad \lambda_c \quad \forall (a_i, \tilde{b}) \in E'_c \tag{1}$$

$$\sum_{e \in \delta'_c(v)} y_e \quad \leq \quad \lambda_c \quad \forall v \in A'_c \cup B'_c \tag{2}$$

$$\sum_{e \in \delta'_c(v)} y_e \; = \; \lambda_c \quad \forall v \in T'_c \qquad \text{and} \qquad y_e \quad \geq \quad 0 \quad \forall e \in E'_c. \tag{3}$$

Constraints (1)-(3) with 1 replacing λ_c (wherever λ_c occurs) describe the stable matching polytope $\mathcal{S}'_c$ of G'_c (by [29]). The stability constraint for any edge $(a_i, \tilde{b})$ in E'_c is given by (1) with 1 replacing λ_c. The stability constraint for edge $(a_{i-1}, d_i(a))$ (resp., $(a_i, d_i(a))$) is given by $\sum_{e \in \delta'_c(v)} y_e = 1$ with $v = a_{i-1}$ (resp., $v = d_i(a)$). Note that both a_{i-1} and $d_i(a)$ are in T'_c.

By Lemma 14, the constraints formulating $\mathcal{S}'_c$ along with $y_{(a,b)} = \sum_i y_{(a_i, \tilde{b})}$ for $(a, b) \in E_c$ describe an extension of the convex hull $\mathcal{F}^0_c$ of the edge incidence vectors of valid matchings in G_c with a witness γ such that $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$.

For any vertex v in G''_c, let $\delta''_c(v)$ be the set of edges incident to v in G''_c and for any neighbor u of v, let $\{w \succ_v u\}$ be the set of all neighbors of v in G''_c that v prefers to u. Let T''_c be the set of vertices in G''_c matched in any stable matching in this graph. Consider constraints (4)-(6) in variables z_e where $e \in E''_c$ and λ_c.

$$\sum_{w\colon w \succ_{a_i} \tilde{b}} z_{(a_i, w)} + \sum_{s\colon s \succ_{\tilde{b}} a_i} z_{(s, \tilde{b})} + z_{(a_i, \tilde{b})} \quad \geq \quad 1 - \lambda_c \quad \forall (a_i, \tilde{b}) \in E''_c \tag{4}$$

$$\sum_{e \in \delta''_c(v)} z_e \quad \leq \quad 1 - \lambda_c \quad \forall v \in A''_c \cup B''_c \tag{5}$$

$$\sum_{e \in \delta''_c(v)} z_e \; = \; 1 - \lambda_c \quad \forall v \in T''_c \qquad \text{and} \qquad z_e \quad \geq \quad 0 \qquad \forall e \in E''_c \tag{6}$$

Constraints (4)–(6) with 1 replacing $1 - \lambda_c$ (wherever $1 - \lambda_c$ occurs) describe the stable matching polytope $\mathcal{S}''_c$ of G''_c (by [29]). The stability constraint for $(a_i, \tilde{b}) \in E''_c$ is given by (4) with 1 replacing $1 - \lambda_c$; the stability constraint for edge $(a_{i-1}, d_i(a))$ (resp., $(a_i, d_i(a))$) is given by $\sum_{e \in \delta''_c(v)} z_e = 1$ with $v = a_{i-1}$ (resp., $v = d_i(a)$). Both a_{i-1} and $d_i(a)$ are in T''_c.

By Lemma 15, the constraints formulating $\mathcal{S}''_c$ along with $z_{(a,b)} = \sum_i z_{(a_i,\tilde{b})}$ for $(a,b) \in E_c$ describe an extension of the convex hull $\mathcal{F}^1_c$ of the edge incidence vectors of valid matchings in G_c with a witness γ such that $\gamma_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\}$ for all $v \in C$.

We know from Lemma 12 that any valid matching in C has a witness γ where either (i) $\gamma_v \in \{0, \ldots, \pm 2k\}$ for all $v \in C$ or (ii) $\gamma_v \in \{\pm 1, \ldots, \pm(2k+1)\}$ for all $v \in C$. So the convex hull of $\mathcal{F}^0_c \cup \mathcal{F}^1_c$ is the valid matching polytope $\mathcal{F}_c$ of G_c. Consider constraints (7)-(8).

$$x_{(a,b)} \quad = \quad \sum_i y_{(a_i,\tilde{b})} + \sum_i z_{(a_i,\tilde{b})} \quad \forall (a,b) \in E_c \tag{7}$$

$$x_e \quad = \quad 0 \;\; \forall e \in (E \cap (C \times C)) \setminus E_c \qquad \text{and} \qquad 0 \le \lambda_c \le 1 \tag{8}$$

The summations over i in constraint (7) are over appropriate i, i.e., if a and b are in S then $x_{(a,b)} = \sum_{i=-k}^{k} y_{(a_i,\tilde{b})} + \sum_{i=-k}^{k+1} z_{(a_i,\tilde{b})}$. If a is in U then $x_{(a,b)} = y_{(a_0,\tilde{b})} + \sum_{i=-k}^{1} z_{(a_i,\tilde{b})}$ and if b is in U then $x_{(a,b)} = y_{(a_0,\tilde{b})} + \sum_{i=0}^{k+1} z_{(a_i,\tilde{b})}$.

Using Balas' theorem [2] to formulate an extension of the convex hull of $\mathcal{F}^0_c \cup \mathcal{F}^1_c$ introduces the variable $\lambda_c \in [0,1]$ and we get constraints (1)-(8) as given above. Thus the polytope defined by (1)-(8) is an extension of the polytope $\mathcal{F}_c$. Hence Theorem 16 follows.

▶ **Theorem 16.** *The polytope $\mathcal{P}_c$ defined by constraints* (1)-(8) *is an extension of the convex hull $\mathcal{F}_c$ of edge incidence vectors of valid matchings in G_c.*

For any two distinct connected components C and C' in G_p, the variables in the formulation of $\mathcal{P}_c$ and those in the formulation of $\mathcal{P}_{c'}$ are distinct. By listing the constraints in the formulation of $\mathcal{P}_c$ over all the non-trivial connected components C in G_p (i.e., $|C| \ge 2$) along with $x_e = 0$ for $e \in E \setminus \cup_C E_c$ (where the union is over all the non-trivial connected components C in G_p), we obtain a compact extended formulation for the fairly popular matching polytope of G. Linear programming on this formulation finds a min-cost fairly popular matching in G in polynomial time. This proves Theorem 4 stated in Section 1.

4 A Hardness Result

We prove Proposition 7 and Theorem 8 in this section. Let $\mathcal{M}_G$ be the matching polytope of the bipartite graph $G = (A \cup B, E)$ where $|A \cup B| = n$ and $|E| = m$. The polytope $\mathcal{M}_G \subseteq \mathbb{R}^m$ is described by the following constraints:

$$\sum_{e \in \delta(v)} x_e \le 1 \;\; \forall v \in A \cup B \qquad \text{and} \qquad x_e \ge 0 \;\; \forall e \in E.$$

For any vertex v, $\delta(v)$ is the set of edges in E incident to v. Any point $x \in \mathcal{M}_G$ is a fractional matching. Let $\tilde{E} = E \cup \{(v,v) : v \in A \cup B\}$ and let $\tilde{G} = (A \cup B, \tilde{E})$. That is, $\tilde{G}$ has self-loops (v,v) for all $v \in A \cup B$. The interpretation is that every vertex v is its own last choice neighbor. So we can regard any fractional matching x as a *perfect* fractional matching in $\tilde{G}$ by setting $x_{(v,v)} = 1 - \sum_{e \in \delta(v)} x_e$ for all vertices v.

For any matching M, recall the edge weight function wt_M defined in Section 3. This was defined in the graph $G_p = (A \cup B, \tilde{E}_p)$ and it easily extends (by the same definition) to $\tilde{G} = (A \cup B, \tilde{E})$. For any edge $e \in E$, $\mathsf{wt}_M(e) \in \{0, \pm 2\}$ and for any self-loop (v,v), $\mathsf{wt}_M(v,v) \in \{0, -1\}$. For any fractional matching x:

$$\Delta(x, M) \;\; = \;\; \mathsf{wt}_M(x) \;\; = \;\; \sum_{e \in \tilde{E}} \mathsf{wt}_M(e) \cdot x_e.$$

As shown in [26], this is exactly the same as defining $\Delta(x, M) = \Delta(\Pi, M)$ where Π is any mixed matching that is equivalent to x. Any popular matching M satisfies $\Delta(x, M) \le 0$ for all $x \in \mathcal{M}_G$. Note that the constraint $\Delta(x, M) \le 0$ involves $m + n$ variables x_e for $e \in \tilde{E}$. By substituting $x_{(v,v)} = 1 - \sum_{e\in\delta(v)} x_e$ for every vertex v, this constraint involves only the m variables x_e for $e \in E$.

▶ **Observation 17.** *Let $\mathcal{X} \subseteq \mathbb{R}^m$ be the convex hull of the edge incidence vectors of matchings that are not defeated by any popular matching. The polytope $\mathcal{X}$ is a face of $\mathcal{M}_G$.*

Proof. Every $x \in \mathcal{M}_G$ satisfies $\Delta(x, N) \le 0$ for all popular matchings N. So the intersection of $\mathcal{M}_G$ with the constraints $\Delta(x, N) = 0$ for all popular matchings N is a face $\mathcal{Q}$ of $\mathcal{M}_G$. The polytope $\mathcal{Q}$ is integral and every integral point in $\mathcal{Q}$ is the edge incidence vector of a matching not defeated by any popular matching. Moreover, the edge incidence vector of every matching that is not defeated by any popular matching is in $\mathcal{Q}$. Hence $\mathcal{Q} = \mathcal{X}$. ◀

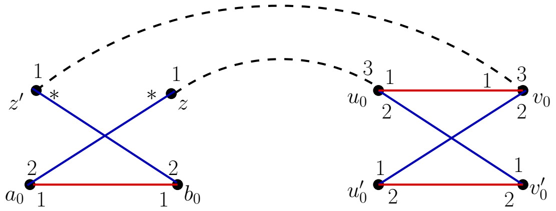
The following constraints in the variables x_e for $e \in E$ describe the polytope $\mathcal{X}$:

$$\Delta(x, N) \;=\; 0 \;\forall \text{ popular matchings } N, \quad \sum_{e\in\delta(v)} x_e \le 1 \quad \forall\, v \in A\cup B, \quad \text{and} \quad x_e \;\ge\; 0 \;\forall\, e \in E.$$

There are exponentially many constraints here. However, $\mathcal{X}$ is a polytope in $\mathbb{R}^m$ and so at most m of the tight constraints $\Delta(x, N) = 0$ are necessary and the rest are redundant. Thus there exist at most $k \le m$ popular matchings $N_1, \ldots, N_k$ such that if a matching M satisfies $\Delta(M, N_i) = 0$ for $1 \le i \le k$ then the edge incidence vector of M belongs to $\mathcal{X}$, i.e., such a matching M is not defeated by any popular matching. Hence Proposition 7 follows.

The NP-hardness proof. We now prove Theorem 8 which states that in spite of the compactness result given by Proposition 7, it is NP-complete to decide if there exists a popular matching that defeats a given matching M. The reduction is from 1-in-3 SAT. This is the set of 3CNF formulas where each clause has 3 literals, none negated, such that there is a satisfying assignment that makes exactly one literal true in each clause.

Given such an input formula ψ, to decide if ψ is 1-in-3 satisfiable is NP-complete [30]. Given ψ, as done in [10], we will construct an instance G described below. The graph G has several gadgets. We are interested in two particular gadgets illustrated in Figure 2. These are on the 8 vertices: $a_0, z', u_0, u_0' \in A$ and $b_0, z, v_0, v_0' \in B$.



■ **Figure 2** The numbers on edges denote preferences: 1 is top choice, 2 is second choice, and 3 is third choice; $*$ denotes a number > 1. The red edges are present in all stable matchings and the blue edges are present in all max-size popular matchings in G.

The top choices of z and z' are u_0 and v_0, respectively. However (z, u_0) and (z', v_0) (the dashed edges in Figure 2) do not belong to any popular matching. The vertices z, z' are adjacent to many vertices in the rest of the graph: we refer to [10] for these details – it is these vertices in the rest of the graph that represent the given formula ψ.

Let P be any popular matching in G. It was shown in [10] that P contains either (a_0, b_0) or the pair $(a_0, z), (z', b_0)$. Also P contains either the pair $(u_0, v_0), (u'_0, v'_0)$ or the pair $(u_0, v'_0), (u'_0, v_0)$. No other edge incident to any of these 8 vertices in Figure 2 belongs to any popular matching in G. The following hardness result [10, Theorem 4.2] will be crucial.

▶ **Theorem 18** ([10]). *The instance G has a popular matching that contains the three edges $(u_0, v'_0), (u'_0, v_0)$, and (a_0, b_0) if and only if ψ is 1-in-3 satisfiable.*

We will use the above instance $G = (A \cup B, E)$ to show the NP-hardness of deciding if there exists a popular matching that defeats a given matching M. Let $M = M_0 \cup M_1$ where $M_1 = \{(a_0, b_0), (u_0, z), (z', v_0), (u'_0, v'_0)\}$ and M_0 is any stable matching in the subgraph induced on $(A \cup B) \setminus S$, where $S = \{a_0, b_0, z', z, u_0, v_0, u'_0, v'_0\}$.

▶ **Lemma 19.** *There exists a popular matching in G that defeats M if and only if ψ is 1-in-3 satisfiable.*

Proof. Let G_1 be the subgraph of G induced on S and let G_0 be the subgraph induced on $(A \cup B) \setminus S$, where $S = \{a_0, b_0, z, z', u_0, v_0, u_1, v_1\}$.

The ⇒ direction. Suppose there is a popular matching N that is more popular than M. No edge between G_0 and G_1 belongs to any popular matching [10], hence $N = N_0 \cup N_1$, where N_i is within G_i, for $i = 0, 1$. Since N is popular in G, the matchings N_0 and N_1 have to be popular in G_0 and G_1, respectively.

We have $\Delta(N, M) = \Delta(N_0, M_0) + \Delta(N_1, M_1)$. Since $\Delta(N, M) > 0$ and $\Delta(N_0, M_0) = 0$ (because M_0 and N_0 are popular matchings in G_0), it follows that $\Delta(N_1, M_1) > 0$.

The graph G_1 has three popular matchings and only one of them defeats M_1. This is the matching $\{(u_0, v'_0), (u'_0, v_0), (a_0, b_0)\}$ that leaves z, z' unmatched. It is easy to check that the other popular matchings in G_1 – these are $P = \{(a_0, b_0), (u_0, v_0), (u'_0, v'_0)\}$ and $P' = \{(a_0, z), (z', b_0), (u_0, v'_0), (u'_0, v_0)\}$ – do not defeat M_1.

So $N_1 = \{(u_0, v'_0), (u'_0, v_0), (a_0, b_0)\}$. We have $\Delta(N_1, M_1) = 4 - 2 = 2$ since u_0, v_0, u'_0, v'_0 prefer N_1 to M_1 while z, z' prefer M_1 to N_1 and a_0, b_0 are indifferent between N_1 and M_1. Since $N_1 \subseteq N$, it follows that N is a popular matching in G that contains $(u_0, v'_0), (u'_0, v_0)$, and (a_0, b_0). This means that ψ is 1-in-3 satisfiable (by Theorem 18).

The ⇐ direction. Suppose ψ is 1-in-3 satisfiable. Then we know from Theorem 18 that there is a popular matching P that contains the edges $(u_0, v'_0), (u'_0, v_0), (a_0, b_0)$. We claim that $\Delta(P, M) > 0$. Let us partition P into $P_0 \cup P_1$ where $P_1 = \{(u_0, v'_0), (u'_0, v_0), (a_0, b_0)\}$ and $P_0 = P \setminus P_1$. We have $\Delta(P, M) = \Delta(P_1, M_1) + \Delta(P_0, M_0)$.

Observe that $\Delta(P_1, M_1) = 4 - 2 = 2$. Moreover, $\Delta(P_0, M_0) = 0$ by the popularity of P_0 and M_0 in G_0. So $\Delta(P, M) = 2$, i.e., the popular matching P defeats M. ◀

Lemma 19 shows that it is NP-hard to decide if there exists a popular matching that defeats a given matching M. This problem is NP-complete since a "yes"-instance M has a popular matching (which is easy to verify [4, 18]) that defeats it. Thus Theorem 8 stated in Section 1 follows.

5 Conclusions

We introduced a relaxation of popular matchings called *fairly popular* matchings in a marriage instance $G = (A \cup B, E)$. Unlike popular matchings, fairly popular matchings may lose to other matchings; however any matching N that defeats a fairly popular matching M does not belong to the support of any popular mixed matching, thus such a matching N can be considered to be quite *far* from being popular. So there is no “viable alternative” that defeats a fairly popular matching. Hence fairly popular matchings are a meaningful generalization of popular matchings.

We characterized matchings that belong to the support of popular mixed matchings. We showed that a matching M belongs to the support of a popular mixed matching if and only if M is undefeated by popular mixed matchings. We also gave a combinatorial characterization of such matchings. This allowed us to characterize fairly popular matchings in terms of witnesses and to use the stable matching machinery to formulate a compact extension of the fairly popular matching polytope. Thus the min-cost fairly popular matching problem can be solved in polynomial time. We also showed that it is NP-complete to decide if there exists a popular matching that is more popular than a given matching M.

References

1 A. Abdulkadiroğlu and T. Sönmez. School choice: a mechanism design approach. *American Economic Review*, 93(3):729–747, 2003.

2 E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

3 S. Baswana, P. P. Chakrabarti, S. Chandran, Y. Kanoria, and U. Patange. Centralized admissions for engineering colleges in India. *INFORMS Journal on Applied Analytics*, 49(5):338–354, 2019.

4 P. Biro, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of the 7th International Conference on Algorithms and Complexity (CIAC)*, pages 97–108, 2010.

5 Canadian Resident Matching Service. How the matching algorithm works. `http://carms.ca/algorithm.htm`.

6 Á. Cseh. Popular matchings. Trends in Computational Social Choice, Ulle Endriss (ed.), 2017.

7 Á. Cseh and T. Kavitha. Popular edges and dominant matchings. *Mathematical Programming*, 172(1):209–229, 2018.

8 L. Ehlers and T. Morrill. (Il)legal assignments in school choice. *The Review of Economic Studies*, 87(4):1837–1875, 2020.

9 Y. Faenza and T. Kavitha. Quasi-popular matchings, optimality, and extended formulations. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 325–344, 2020.

10 Y. Faenza, T. Kavitha, V. Powers, and X. Zhang. Popular matchings and limits to tractability. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2790–2809, 2019.

11 T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45(2):233–284, 1992.

12 T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319, 1994.

13 T. Fleiner. A fixed-point approach to stable matchings and some applications. *Mathematics of Operations Research*, 28(1):103–126, 2003.

14 D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.

15 D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.

16 P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

17 D. Gusfield and R. W. Irving. *The stable marriage problem: Structure and algorithms.* MIT Press, 1989.

18 C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 222:180–194, 2013.

19 C.-C. Huang and T. Kavitha. Popularity, mixed matchings, and self-duality. *Mathematics of Operations Research*, 46(2):405–427, 2021.

20 R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 34(3):532–543, 1987.

21 T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43(1):52–71, 2014.

22 T. Kavitha. Popular half-integral matchings. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 22:1–22:13, 2016.

23 T. Kavitha. Matchings, critical nodes, and popular solutions. In *Proceedings of the the 41st Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 25:1–25:19, 2021.

24 T. Kavitha. Maximum matchings and popularity. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 85:1–85:21, 2021.

25 T. Kavitha. Min-cost popular matchings. In *Proceedings of the the 40th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 25:1–25:17, 2021.

26 T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412:2679–2690, 2011.

27 M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of the 8th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 593–604, 2008.

28 National Resident Matching Program. Why the Match? `http://www.nrmp.org/whythematch.pdf`.

29 U. G. Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67, 1992.

30 T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

31 C.-P. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998.

32 E. G. Thurber. Concerning the maximum number of stable matchings in the stable marriage problem. *Discrete Mathematics*, 248(1-3):195–219, 2002.

33 J. H. Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

A Appendix: Missing Proofs

We prove Lemma 10 here. Before we prove this lemma, we discuss some preliminaries that will be used in this proof. Let $\tilde{G} = (A \cup B, \tilde{E})$ be the graph G augmented with self-loops at all vertices. So each vertex v regards itself as its last choice neighbor and any matching M in G becomes a perfect matching $\tilde{M}$ in $\tilde{G}$ by augmenting M with self-loops at vertices left unmatched in M.

For any matching M, recall the edge weight function wt_M defined at the start of Section 3 in G_p. We now extend this edge weight function to all edges e in G, so $\mathsf{wt}_M(e) \in \{\pm 2, 0\}$ where $\mathsf{wt}_M(e) = 2$ if e blocks M and so on. For any vertex v, let $\mathsf{wt}_M(v,v) = 0$ if v is left unmatched in M, else $\mathsf{wt}_M(v,v) = -1$.

For any matching N in G, we have $\mathsf{wt}_M(\tilde{N}) = \Delta(N, M)$. So M is popular in G if and only if $\mathsf{wt}_M(\tilde{N}) \le 0$ for all matchings N. Since $\mathsf{wt}_M(\tilde{M}) = 0$, the matching M is popular in G if and only if the optimal value of (LP3) is 0. The linear program (LP4) is the dual LP.

▶ **Theorem 20** ([22, 26]). *A matching M in $G = (A \cup B, E)$ is popular if and only if there exists $y \in \{0, \pm 1\}^n$ such that $\sum_{v \in A \cup B} y_v = 0$ along with $y_a + y_b \ge \mathsf{wt}_M(a,b)$ for all $(a,b) \in E$ and $y_v \ge \mathsf{wt}_M(v,v)$ for all $v \in A \cup B$.*

$$\begin{aligned} \max \sum_{e\in\tilde{E}} \mathsf{wt}_M(e)\cdot x_e \qquad & \text{(LP3)} \\ \text{s.t.} \quad \sum_{e\in\delta(v)\cup\{(v,v)\}} x_e = 1 \quad & \forall\, v\in A\cup B \\ x_e \geq 0 \quad & \forall\, e\in\tilde{E}. \end{aligned}$$

$$\begin{aligned} \min \sum_{v\in A\cup B} y_v \qquad & \text{(LP4)} \\ \text{s.t.} \quad y_a + y_b \geq \mathsf{wt}_M(a,b) \quad & \forall\,(a,b)\in E \\ y_v \geq \mathsf{wt}_M(v,v) \quad & \forall\, v\in A\cup B. \end{aligned}$$

We will call a vector y, as given in Theorem 20, a *dual certificate* for popular matching M. Note that 0 is a dual certificate for any stable matching since a matching M is stable if and only if $\mathsf{wt}_M(e) \leq 0$ for all edges e.

We need to show in Lemma 10 that any matching in $H = (A_H \cup B_H, E_H)$ (see Figure 1) not defeated by any popular matching contains only popular edges. Every popular matching in H is perfect (since H has a perfect stable matching). As shown in [7], in such a case, there is a surjective map from the set of stable matchings in an auxiliary instance $H' = (A'_H \cup B'_H, E'_H)$ to the set of popular matchings in H. The graph H' is defined as follows:

- $A'_H = \{a, a' : a \in A_H\}$. So every $a \in A_H$ has two copies a and a' in A'_H.
- $B'_H = B_H \cup \{d(a) : a \in A_H\}$. So for every $a \in A_H$, there is a dummy vertex $d(a)$ in B'_H.

The vertex $d(a)$ has only two neighbors a, a' and $d(a)$ prefers a to a'. Every $(a,b) \in E_H$ has two copies (a,b) and (a',b) in E'_H. For any $a \in A_H$, if a's preference order in H is $b_1 \succ \cdots \succ b_r$ then a's preference order in H' is $b_1 \succ \cdots \succ b_r \succ d(a)$ and a''s preference order in H' is $d(a) \succ b_1 \succ \cdots \succ b_r$.

Let $b \in B_H$. If b's preference order in H is $a_1 \succ \cdots \succ a_k$ then b's preference order in H' is $a'_1 \succ \cdots \succ a'_k \succ a_1 \succ \cdots \succ a_k$, i.e., all its primed neighbors followed by all its unprimed neighbors, where the order among primed/unprimed neighbors is b's original order in H.

Let M' be any stable matching in H'. Then M' maps to the following matching in H: $M = \{(a,b) : (a,b) \text{ or } (a',b) \text{ is in } M'\}$.

For each $a \in A_H$, note that the stable matching M' has to match one of a, a' to $d(a)$ since $d(a)$ is the top choice neighbor for a'. The matching M is popular in H since it has the following witness $y \in \{\pm 1\}^{n_H}$: (where $|A_H \cup B_H| = n_H$)

1. for $a \in A_H$: if $(a', d(a)) \in M'$ then $y_a = 1$; else $y_a = -1$.
2. for $b \in B_H$: if b's partner in M' is a *primed* vertex (such as a') then $y_b = 1$; else $y_b = -1$.

We refer to [7, 19] for the details that y is a feasible solution to (LP2) and $\sum_{v\in A_H\cup B_H} y_v = 0$.

Proof of Lemma 10. Let N be a matching in H that contains an *unpopular* edge (s,t). We will now show there is a popular matching in H that defeats N. Call an edge e *stable* if there is a stable matching in H that contains e. The following result on stable matchings in a marriage instance will be useful to us.

▶ **Proposition 21** ([17, proof of Lemma 2.5.1]). *Suppose (s,t_0) and (s,t_1) are stable edges while (s,t) is not a stable edge where $t_1 \succ_s t \succ_s t_0$. Then there is a stable matching M where both s and t prefer their respective partners in M to each other.*

Let t_ℓ be the partner of s in the A_H-optimal stable matching M_ℓ in H and let t_r be the partner of s in the B_H-optimal stable matching M_r in H.

Case 1. Suppose $t_\ell \succ_s t \succ_s t_r$. Since the edge (s,t) is not stable while (s,t_ℓ) and (s,t_r) are stable edges, there is a stable matching M in H such that both s and t prefer their partners in M to each other (by Proposition 21). So $\mathsf{wt}_M(s,t) = -2$. This makes the edge (s,t) *slack* wrt to the popular matching M and its witness $y = 0$, i.e., $\mathsf{wt}_M(s,t) = -2 < 0 = y_s + y_t$.

Since $y = 0$ is a feasible solution to (LP2), $\mathsf{wt}_M(\tilde{N}) = \sum_{e \in \tilde{N}} \mathsf{wt}_M(e) < \sum_v y_v = 0$ (since $\mathsf{wt}_M(s,t) < y_s + y_t$). Thus $\Delta(N, M) < 0$, i.e., the stable matching M defeats N.

Case 2. Suppose $t \succ_s t_\ell$. That is, s prefers t to its most preferred stable partner t_ℓ in H. Consider the following two stable matchings in $H' = (A'_H \cup B'_H, E'_H)$:

$$M'_r \quad = \quad \{(a,b) : (a,b) \in M_r\} \cup \{(a', d(a)) : a \in A_H\}$$

$$M'_\ell \quad = \quad \{(a',b) : (a,b) \in M_\ell\} \cup \{(a, d(a)) : a \in A_H\}.$$

The vertex s' is matched to its top choice neighbor $d(s)$ in M'_r and it is matched to t_ℓ in M'_ℓ. Recall that $d(s) \succ_{s'} t \succ_{s'} t_\ell$. Since (s,t) is not a popular edge in H, the edge (s',t) is not stable in H'. We know that $(s', d(s))$ and (s', t_ℓ) are stable edges in H', hence there exists a stable matching M' in H' such that both s' and t prefer their respective partners in M' to each other (by Proposition 21). Observe that t's partner in M' has to be a *primed* neighbor (call it v') since t cannot prefer an *unprimed* neighbor to s'. So M' contains edges (s', u) and (v', t) where s' and t prefer their respective partners (u and v') to each other.

Let the stable matching M' in H' map to the popular matching M in H; let $y \in \{\pm 1\}^{n_H}$ be M's witness as described earlier. There are two subcases here.

- The vertex $u = d(s)$. So M' contains (s,b) (for some $b \in B_H$) and (v', t) where t prefers v' to s', i.e., t prefers v to s. The edges $(s,b), (v,t)$ are in M, where $\mathsf{wt}_M(s,t) \le 0$. We have $y_s = y_t = 1$ (by 1. and 2. stated earlier). Hence $\mathsf{wt}_M(s,t) \le 0 < 2 = y_s + y_t$.
- The vertex $u \ne d(s)$. So M' contains (s', u) and (v', t) where s prefers u to t and similarly, t prefers v to s. The edges $(s,u), (v,t)$ are in M and $\mathsf{wt}_M(s,t) = -2$. We have $y_s = -1$ and $y_t = 1$ (by 1. and 2. stated earlier). Hence $\mathsf{wt}_M(s,t) = -2 < 0 = y_s + y_t$.

So in both cases, the edge (s,t) is slack wrt M and its witness y. So complementary slackness (the same argument as given in case 1) implies that $\Delta(N, M) < 0$, i.e., the popular matching M defeats N.

Case 3. The last case is $t_r \succ_s t$. So s prefers its least preferred stable partner to t. If t also prefers its partner in M_r to s then M_r is a stable matching where both s and t prefer their respective partners to each other. This implies that M_r defeats N.

Else t prefers s to its partner in M_r, i.e., t prefers s to its most preferred stable partner. Observe that this is exactly the same as case 2 with the roles of s and t swapped. Thus an analogous argument shows that H has a popular matching that defeats N.

Proof of Lemma 14. Let M_c be a valid matching in G_c with a witness γ such that $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\}$ for all $v \in C$. Recall that S (resp., U) is the set of stable (resp., unstable) vertices in G. We claim that all vertices in $S \cap C$ are matched in M_c and no vertex in $U \cap C$ is matched in M_c.

Consider (LP1) with M_c replacing M and $\tilde{E}_c = \tilde{E}_p \cap (C \times C)$ replacing $\tilde{E}_p$. The optimal value of this LP is 0 since there exists a dual feasible solution γ with $\sum_{u \in C} \gamma_u = 0$ (recall that γ obeys properties 1-3). Let N be a stable matching in G and let $N_c = N \cap (C \times C)$. If M_c leaves a vertex $v \in S \cap C$ unmatched then $\Delta(N_c, M_c) > 0$ (as shown in [18]), a contradiction to the optimal value of (LP1) being 0. Thus M_c matches all vertices in $S \cap C$. Since the self-loop $(u,u) \in \tilde{N}_c$ for any $u \in U \cap C$, the constraint $\gamma_u \ge \mathsf{wt}_{M_c}(u,u)$ is tight (by complementary slackness). Because $\mathsf{wt}_{M_c}(u,u) \in \{0, -1\}$ and γ_u is even, it follows that $\gamma_u = \mathsf{wt}_{M_c}(u,u) = 0$, i.e., u is left unmatched in M_c.

We need to show there is no blocking edge with respect to M'_c and this proof is similar to a proof in [24] on popular perfect matchings. Any dummy vertex $d_i(a)$ is matched either to its top choice neighbor a_{i-1} or to its second choice neighbor a_i; in the latter case, its top choice neighbor a_{i-1} is matched to a more preferred neighbor. Thus no blocking edge is incident to any dummy vertex. Let us now show that no blocking edge is incident to any other vertex in G'_c. Observe that $\tilde{M}_c$ is an optimal solution to (LP1), so for any $(p,q) \in M_c$, we have $\gamma_p + \gamma_q = \mathsf{wt}_{M_c}(p,q) = 0$ (by complementary slackness).

Let $a \in U \cap C_A$ and let $(a,b) \in E_c$. We have $\gamma_a = 0$ and $\gamma_a + \gamma_b \geq \mathsf{wt}_{M_c}(a,b) \geq 0$. So $\gamma_b \geq 0$. If $\gamma_b = 0$ then $\mathsf{wt}_{M_c}(a,b) = 0$, i.e., $(z_0, \tilde{b}) \in M'_c$ for some neighbor z that b prefers to a. Else $\gamma_b > 0$ and so $(z_i, \tilde{b}) \in M'_c$ for some neighbor z with $-2i = \gamma_z = -\gamma_b < 0$, so $i > 0$, i.e., $\tilde{b}$ is matched to a neighbor in G'_c that it prefers to a_0. Hence $(a_0, \tilde{b})$ does not block M'_c.

Let us now show there is no blocking edge incident to a_ℓ, where $a \in S \cap C_A$ and $-k \leq \ell \leq k$. Suppose $\gamma_a = -2i$ and $(a,w) \in M_c$. Then $(a_i, \tilde{w}) \in M'_c$ and all of $a_{i+1}, \ldots, a_k$ are matched to their respective top choice neighbors $d_{i+1}(a), \ldots, d_k(a)$. Hence there is no blocking edge incident to a_j for $j \geq i+1$.

Let $(a,b) \in E_c$. If $b \in U$ then $\gamma_b = 0$ and $\gamma_a + \gamma_b \geq \mathsf{wt}_{M_c}(a,b) \geq 0$. So $\gamma_a \geq 0$. If $\gamma_a = 0$ then $\mathsf{wt}_{M_c}(a,b) = 0$, i.e., $(a_0, \tilde{w}) \in M'_c$ for some neighbor w that a prefers to b. Else $\gamma_a > 0$ which implies that $i < 0$ and so $(a_0, d_0(a)) \in M_c$. In either case, a_0 prefers its partner in M'_c to $\tilde{b}$, so $(a_0, \tilde{b})$ does not block M'_c.

Let $b \in S$. Since $\gamma_a + \gamma_b \geq \mathsf{wt}_{M_c}(a,b) \geq -2$, it follows that $\gamma_b \geq 2(i-1)$. Thus $(z_j, \tilde{b}) \in M'_c$ where $j \geq i-1$. Hence $\tilde{b}$ prefers its partner in M'_c to all a_j, where $j \leq i-2$. We now show that neither $(a_{i-1}, \tilde{b})$ nor $(a_i, \tilde{b})$ blocks M'_c.

- If $j \geq i+1$ then $\tilde{b}$ prefers its partner z_j in M'_c to both a_i and a_{i-1}. Hence neither $(a_{i-1}, \tilde{b})$ nor $(a_i, \tilde{b})$ blocks M'_c.
- If $j = i$ then $\gamma_a + \gamma_b = -2i + 2i = 0 \geq \mathsf{wt}_{M_c}(a,b)$. Thus either $(a_i, \tilde{b}) \in M'_c$ or one of $a_i, \tilde{b}$ prefers its partner in M'_c to the other. Hence neither $(a_i, \tilde{b})$ nor $(a_{i-1}, \tilde{b})$ blocks M'_c in this case as well.
- If $j = i-1$ then $\gamma_a + \gamma_b = -2i + 2(i-1) = -2 \geq \mathsf{wt}_{M_c}(a,b)$. So $\mathsf{wt}_{M_c}(a,b) = -2$, i.e., both a and b prefer their partners in M_c to each other. Hence $\tilde{b}$ prefers z_{i-1} to a_{i-1} and similarly, a_i prefers $\tilde{w}$ to $\tilde{b}$. Thus in this case also neither $(a_{i-1}, \tilde{b})$ nor $(a_i, \tilde{b})$ blocks M'_c.

We prove the converse now. Let N be any stable matching in G and let $N_c = N \cap (C \times C)$. It is easy to check that $N'_c = \{(a_0, \tilde{b}) : (a,b) \in N_c\} \cup \{(a_i, d_{i+1}(a)) : a \in S \cap C_A \text{ and } i < 0\}$ $\cup \{(a_i, d_i(a)) : a \in S \cap C_A \text{ and } i > 0\}$ is a stable matching in G'_c. The set of vertices left unmatched in N'_c is $\{a_0, \tilde{b} : a, b \in U \cap C\}$. Hence the stable matching M'_c matches all vertices of G'_c except the vertices $a_0, \tilde{b}$, where $a, b \in U \cap C$.

In order to prove that M_c is valid in G_c, we define γ as follows:

- for every vertex $u \in U \cap C$, let $\gamma_u = 0$;
- for every edge $(p_i, \tilde{q}) \in M'_c$, let $\gamma_p = -2i$ and $\gamma_q = 2i$.

Since $-k \leq i \leq k$, it immediately follows that $\gamma_v \in \{0, \pm 2, \ldots, \pm 2k\} \, \forall v \in C$. For any $u \in U \cap C$ (each such vertex is unmatched in M_c), we have $\gamma_u = 0 = \mathsf{wt}_{M_c}(u,u)$. We also have $\sum_{v \in C} \gamma_v = \sum_{(p,q) \in M_c} (\gamma_p + \gamma_q) = 0$.

Thus we are left to show the constraints $\gamma_a + \gamma_b \geq \mathsf{wt}_{M_c}(a,b)$ for all $(a,b) \in E_c$. Then it will follow that properties 1-3 hold and thus M_c is valid in G_c with γ as a witness. Suppose $\gamma_a = -2i$ and $\gamma_b = 2j$. We need to show that $-2i + 2j \geq \mathsf{wt}_{M_c}(a,b)$ and this proof is similar to a proof in [23] on popular critical matchings. Let us consider the following 4 cases:

1. $j \geq i+1$: So $\gamma_a + \gamma_b \geq -2i + 2(i+1) = 2 \geq \mathsf{wt}_{M_c}(a,b)$ since $\mathsf{wt}_{M_c}(e) \in \{0, \pm 2\}$ for any $e \in E$.
2. $j = i$: Since the edge $(a_i, \tilde{b})$ does not block M'_c, either $(a_i, \tilde{b}) \in M'_c$ or one of $a_i, \tilde{b}$ is matched to a neighbor preferred to the other. Recall that the preference order of $\tilde{b}$ among level i neighbors in G'_c is exactly as per its preference order in G. Thus either $(a,b) \in M_c$ or one of a, b is matched in M_c to a neighbor preferred to the other. Hence $\gamma_a + \gamma_b = -2i + 2i = 0 \geq \mathsf{wt}_{M_c}(a,b)$.

3. $j = i-1$: Observe that a has to be a stable vertex, otherwise $i = 0$ and the edge $(a_0, \tilde{b})$ would block M'_c. Since $j \ge -k$, we have $i = j+1 \ge 1-k$; so there is a vertex $d_i(a)$ which (as a_i's top choice) has to be matched in any stable matching in G'_c. Since $(a_i, \tilde{w}) \in M'_c$ for some $w \in B$, it follows that $(a_{i-1}, d_i(a)) \in M'_c$. So a_{i-1} is matched to its worst choice neighbor and because the edge $(a_{i-1}, \tilde{b})$ does not block M'_c, it follows that $(z_{i-1}, \tilde{b}) \in M'_c$ for some neighbor z that b prefers to a. The vertex $\tilde{b}$ prefers a_i to z_{i-1} since higher level neighbors are preferred to lower level neighbors. Since the edge $(a_i, \tilde{b})$ does not block M'_c, it follows that a prefers w to b. Thus both a and b prefer their respective partners in M_c to each other, so $\mathsf{wt}_{M_c}(a,b) = -2 = -2i + 2(i-1) = \gamma_a + \gamma_b$.
4. $j \le i-2$: As argued in the above case, a has to be a stable vertex and $(a_{i-1}, d_i(a)) \in M'_c$. So a_{i-1} is matched to its worst choice neighbor. Either $\tilde{b}$ is unmatched or $(z_j, \tilde{b}) \in M'_c$ for some $j \le i-2$. In either case, M'_c has a blocking edge – a contradiction to its stability. Thus we cannot have $j \le i-2$. ◀

Proof of Lemma 15. The matching M_c has to match all vertices in $S \cap C$, otherwise we have $\Delta(N_c, M_c) > 0$ where N is any stable matching in G and $N_c = N \cap (C \times C)$, contradicting that there is a feasible solution γ to (LP2)[2] with $\sum_{v \in C} \gamma_v = 0$. Thus $\tilde{M}_c$ is feasible solution to (LP1); in fact, it is an optimal solution to (LP1) since $\mathsf{wt}_{M_c}(\tilde{M}_c) = \Delta(M_c, M_c) = 0$. If M_c leaves a vertex v unmatched then $(v,v) \in \tilde{M}_c$ and so by complementary slackness, we have $\gamma_v = \mathsf{wt}_{M_c}(v,v) = 0$. However all the γ-values are odd. Hence M_c matches all vertices in C.

We need to show that M''_c is stable in G''_c. As argued in the proof of Lemma 14, no blocking edge can be incident to any dummy vertex. Let us now show that there is no blocking edge incident to a_ℓ, where $a \in C_A$ and $\ell \ge -k$.

Suppose $(a, w) \in M_c$. Let $\gamma_a = -(2i-1)$. Then $(a_i, \tilde{w}) \in M''_c$ and $(a_j, d_j(a)) \in M''_c$ for $j \ge i+1$. Since a_j is matched to its top choice neighbor $d_j(a)$, there is no blocking edge incident to a_j for $j \ge i+1$.

Let b be any neighbor of a in G_c, i.e., $(a,b) \in E_c$. Then $\gamma_a + \gamma_b \ge \mathsf{wt}_{M_c}(a,b) \ge -2$, so $\gamma_b \ge 2i - 3 = 2(i-1) - 1$. Thus $(z_j, \tilde{b}) \in M''_c$ where $j \ge i-1$. Hence $\tilde{b}$ prefers its partner z_j to all a_ℓ, where $\ell \le i-2$. We now show that neither $(a_{i-1}, \tilde{b})$ nor $(a_i, \tilde{b})$ blocks M''_c.

- If $j \ge i+1$ then $\tilde{b}$ prefers its partner z_j in M''_c to both a_i and a_{i-1}. Hence neither $(a_{i-1}, \tilde{b})$ nor $(a_i, \tilde{b})$ blocks M'_c.
- If $j = i$ then $\gamma_a + \gamma_b = -(2i-1) + (2i-1) = 0 \ge \mathsf{wt}_{M_c}(a,b)$. Thus either $(a_i, \tilde{b}) \in M''_c$ or one of $a_i, \tilde{b}$ prefers its partner in M''_c to the other. Hence neither $(a_i, \tilde{b})$ nor $(a_{i-1}, \tilde{b})$ blocks M''_c in this case.
- If $j = i-1$ then $\mathsf{wt}_{M_c}(a,b) = -2$ and so both a and b prefer their partners in M_c to each other. So $\tilde{b}$ prefers z_{i-1} to a_{i-1} and similarly, a_i prefers $\tilde{w}$ to $\tilde{b}$. Thus in this case also neither $(a_{i-1}, \tilde{b})$ nor $(a_i, \tilde{b})$ blocks M''_c.

We will now prove the converse. We claim that M''_c is a perfect matching in G''_c. Let N be the max-size popular matching in G computed by the algorithm in [21]; N has a dual certificate in $\{0, \pm 1\}^n$ where every matched vertex has ± 1 in its coordinate. The matching $N_c = N \cap (C \times C)$ matches all the vertices in C (recall that $|C| \ge 2$).

We use N's dual certificate restricted to vertices in C (call this β, thus $\beta \in \{\pm 1\}^{|C|}$) to obtain a stable matching N''_c in G''_c. For $a \in C_A$, let $f(a) = (1 - \beta_a)/2$, so $f(a) = 0$ if $\beta_a = 1$, else $f(a) = 1$. Note that $f(a) = 1$ for every $a \in U \cap C_A$ (see [7, 21] for more details). Let $N''_c = \{(a_{f(a)}, \tilde{b}) : (a,b) \in N_c\} \cup \{(a_i, d_{i+1}(a)) : a \in C_A \text{ and } i < f(a)\} \cup \{(a_i, d_i(a)) : a \in S \cap C_A \text{ and } i > f(a)\}$. The stable matching N''_c matches all vertices in G''_c.

[2] This is the LP dual to (LP1) with M_c replacing M and $\tilde{E}_c = \tilde{E}_p \cap (C \times C)$ replacing $\tilde{E}_p$.

Hence the stable matching M''_c is also a perfect matching in G''_c. Thus M_c matches all vertices in C. In order to prove that M_c is a valid matching in G_c, we define γ as follows:

- for every edge $(p_i, \tilde{q}) \in M''_c$, let $\gamma_p = -(2i-1)$ and $\gamma_q = 2i-1$.

Since $-k \leq i \leq k+1$, we have $\gamma_v \in \{\pm 1, \pm 3, \ldots, \pm(2k+1)\} \, \forall v \in C$. We also have $\sum_{v \in C} \gamma_v = \sum_{(p,q) \in M_c} (\gamma_p + \gamma_q) = 0$.

Furthermore, for any $a \in U \cap C_A$, we have $(a_i, \tilde{w}) \in M''_c$ where $-k \leq i \leq 1$ for some neighbor w; thus $\gamma_a = -(2i-1) \geq -1$. Similarly, for any $b \in U \cap C_B$, we have $(z_j, \tilde{b}) \in M''_c$ where $0 \leq j \leq k+1$ for some neighbor z; thus $\gamma_b = 2j - 1 \geq -1$. Hence for any $u \in U \cap C$, we have $\gamma_u \geq -1 = \mathsf{wt}_{M_c}(u,u)$.

Thus we are left to show the constraints $\gamma_a + \gamma_b \geq \mathsf{wt}_{M_c}(a,b)$ for all $(a,b) \in E_c$. Then it will follow that properties 1-3 hold and thus M_c is valid in G_c with γ as a witness. Suppose $\gamma_a = -(2i-1)$ and $\gamma_b = 2j-1$. As done in the proof of Lemma 14, let us consider the following 4 cases:

1. $j \geq i+1$: So $\gamma_a + \gamma_b \geq 2 \geq \mathsf{wt}_{M_c}(a,b)$ since $\mathsf{wt}_{M_c}(e) \in \{0, \pm 2\}$ for any $e \in E$.
2. $j = i$: Since the edge $(a_i, \tilde{b})$ does not block M''_c, either $(a_i, \tilde{b}) \in M''_c$ or one of $a_i, \tilde{b}$ is matched to a neighbor preferred to the other. Thus either $(a,b) \in M_c$ or one of a, b is matched in M_c to a neighbor preferred to the other. So $\gamma_a + \gamma_b = -(2i-1) + 2i - 1 = 0 \geq \mathsf{wt}_{M_c}(a,b)$.
3. $j = i-1$: So $(a_i, \tilde{w})$ and $(z_{i-1}, \tilde{b})$ are in M''_c. The vertex $\tilde{b}$ prefers a_i to z_{i-1} because higher level neighbors are preferred to lower level neighbors. Since the edge $(a_i, \tilde{b})$ does not block M''_c, it follows that a_i prefers $\tilde{w}$ to $\tilde{b}$. Observe that $(a_{i-1}, d(a_i)) \in M''_c$, thus a_{i-1} is matched to its worst choice neighbor $d(a_i)$. Since the edge $(a_{i-1}, \tilde{b})$ does not block M''_c, it follows that $\tilde{b}$ prefers z_{i-1} to a_{i-1}. Thus both a and b prefer their respective partners in M_c to each other, so $\mathsf{wt}_{M_c}(a,b) = -2 = -(2i-1) + 2(i-1) - 1 = \gamma_a + \gamma_b$.
4. $j \leq i-2$: We have $(z_j, \tilde{b}) \in M''_c$ for some $j \leq i-2$. As argued in the previous case, the edge $(a_{i-1}, d(a_i)) \in M''_c$. This means that M''_c has a blocking edge, a contradiction to its stability. Hence this case does not occur. ◀

Covering Many (Or Few) Edges with k Vertices in Sparse Graphs

Tomohiro Koana
Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Christian Komusiewicz
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany

André Nichterlein
Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Frank Sommer
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany

Abstract

We study the following two fixed-cardinality optimization problems (a maximization and a minimization variant). For a fixed α between zero and one we are given a graph and two numbers $k \in \mathbb{N}$ and $t \in \mathbb{Q}$. The task is to find a vertex subset S of exactly k vertices that has value at least (resp. at most for minimization) t. Here, the value of a vertex set computes as α times the number of edges with exactly one endpoint in S plus $1 - \alpha$ times the number of edges with both endpoints in S. These two problems generalize many prominent graph problems, such as Densest k-Subgraph, Sparsest k-Subgraph, Partial Vertex Cover, and Max $(k,n-k)$-Cut.

In this work, we complete the picture of their parameterized complexity on several types of sparse graphs that are described by structural parameters. In particular, we provide kernelization algorithms and kernel lower bounds for these problems. A somewhat surprising consequence of our kernelizations is that Partial Vertex Cover and Max $(k, n-k)$-Cut not only behave in the same way but that the kernels for both problems can be obtained by the same algorithms.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Kernelization, Partial Vertex Cover, Densest k-Subgraph, Max $(k, n-k)$-Cut, Degeneracy

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.42

Related Version *A full version of the paper is available at*: http://arxiv.org/abs/2201.05465

Funding *Tomohiro Koana*: Supported by the Deutsche Forschungsgemeinschaft (DFG), project DiPa, NI 369/21.
Frank Sommer: Supported by the Deutsche Forschungsgemeinschaft (DFG), project EAGR, KO 3669/6-1.

1 Introduction

Fixed-cardinality optimization problems are a well-studied class of graph problems where one seeks for a given graph G, a vertex set S of size k such that S optimizes some objective function $\mathrm{val}_G(S)$ [6, 8, 7, 26]. Prominent examples of these problems are Densest k-Subgraph [5, 15, 26], Sparsest k-Subgraph [18, 19, 33], Partial Vertex Cover [1, 17, 21], and Max $(k,n-k)$-Cut [7, 31, 32].

A common thread in these examples is that all these problems are formulated in terms of the number of edges that have one or two endpoints in S: In the decision version of Densest k-Subgraph we require that there are at least t edges with both endpoints in S; Clique is the special case where $t = \binom{k}{2}$. Conversely, in Sparsest k-Subgraph we require that at

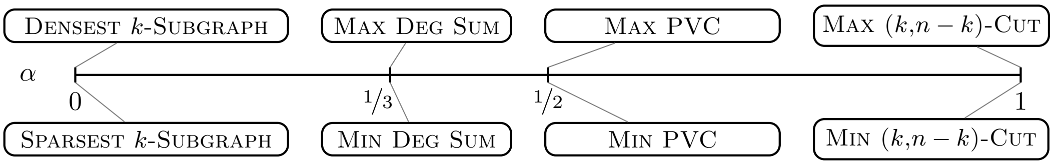


Figure 1 Problem definition cheat sheet.

most t edges have both endpoints in S and Independent Set is the special case with $t = 0$. In Partial Vertex Cover we require that at least t edges have at least one endpoint in S. Finally, in Max $(k,n-k)$-Cut we require that at least t edges have exactly one endpoint in S.

We study the following general problem first defined by Bonnet et al. [3] that contains all of the above problems as special case.[1]

Max α-Fixed Cardinality Graph Partitioning (Max α-FCGP)
Input: A graph G, $k \in \mathbb{N}$, and $t \in \mathbb{Q}$.
Question: Is there a set S of exactly k vertices such that

$$\mathrm{val}(S) := (1-\alpha) \cdot m(S) + \alpha \cdot m(S, V(G) \setminus S) \geq t\ ?$$

Here, $\alpha \in [0,1]$, and $m(S)$ denotes the number of edges with two endpoints in S and $m(S, V(G) \setminus S)$ denotes the number of edges with exactly one endpoint in S. Naturally, one may also consider the minimization problem, denoted as Min α-Fixed Cardinality Graph Partitioning (Min α-FCGP), where we are looking for a set S such that $\mathrm{val}(S) \leq t$.

The value of α describes how strongly edges with exactly one endpoint in S influence the value of S relative to edges with two endpoints in S. For $\alpha = 1/3$, edges with two endpoints in S count twice as much as edges with one endpoint in S and, thus, every vertex contributes exactly its degree to the value of S. Hence, in this case, we simply want to find a vertex set with a largest or smallest degree sum.

More importantly, Max α-FCGP and Min α-FCGP contain all of the above-mentioned problems as special cases (see Figure 1). For example, Partial Vertex Cover (MaxPVC) is Max α-FCGP with $\alpha = 1/2$ as all edges with at least one endpoint in S count the same. Max $(k, n-k)$-Cut is Max α-FCGP with $\alpha = 1$ since edges with both endpoints in S are ignored. Sparsest k-Subgraph is Min α-FCGP with $\alpha = 0$ as only the edges with both endpoints in S count. Consequently, there exist values of α such that Max α-FCGP and Min α-FCGP are NP-hard and W[1]-hard on general graphs with respect to the natural parameter k [7, 9, 12, 17]. This hardness makes it interesting to study these problems on input graphs with special structure and Bonnet et al. [3] and Shachnai and Zehavi [32] studied this problem on bounded-degree graphs.

We continue this line of research and give a complete picture of the parameterized complexity of Min α-FCGP and Max α-FCGP on several types of sparse graphs that are described by structural parameters. In particular, we provide kernelization algorithms and kernel lower bounds for these problems, see Figure 2 for an overview.

[1] On the face of it, the definition of Bonnet et al. [3] seems to be more general as it has separate weight parameters for the internal and outgoing edges. It can be reduced to our formulation by adapting the value of t and thus our results also hold for this formulation.

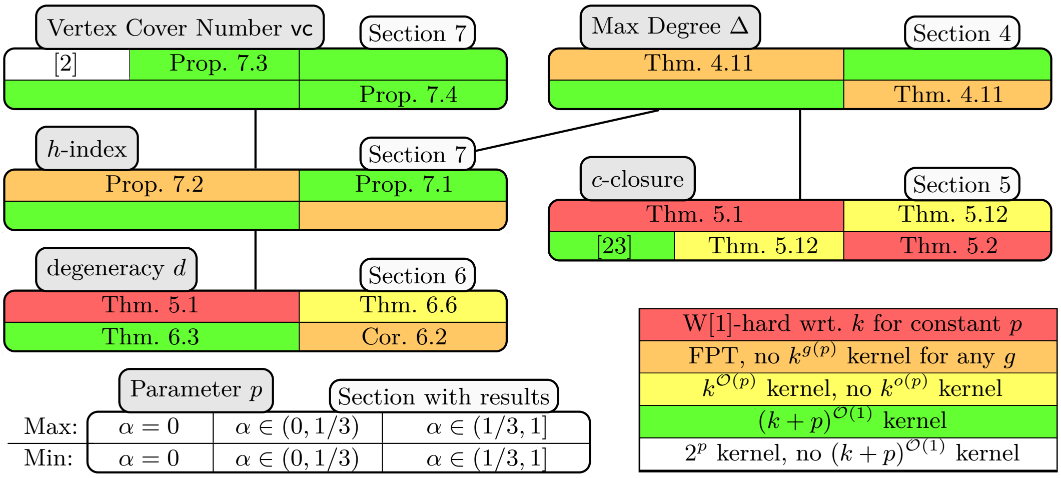

Vertex Cover Number vc
Section 7
[2]
Prop. 7.3
Prop. 7.4
Max Degree Δ
Section 4
Thm. 4.11
Thm. 4.11
h-index
Section 7
Prop. 7.2
Prop. 7.1
c-closure
Section 5
Thm. 5.1
Thm. 5.12
[23]
Thm. 5.12
Thm. 5.2
degeneracy d
Section 6
Thm. 5.1
Thm. 6.6
Thm. 6.3
Cor. 6.2
W[1]-hard wrt. k for constant p
FPT, no $k^{g(p)}$ kernel for any g
$k^{\mathcal{O}(p)}$ kernel, no $k^{o(p)}$ kernel
$(k+p)^{\mathcal{O}(1)}$ kernel
2^p kernel, no $(k+p)^{\mathcal{O}(1)}$ kernel
Parameter p
Section with results
Max: $\alpha = 0$ | $\alpha \in (0, 1/3)$ | $\alpha \in (1/3, 1]$
Min: $\alpha = 0$ | $\alpha \in (0, 1/3)$ | $\alpha \in (1/3, 1]$

Figure 2 Overview over our results. Each box displays the parameterized results (see also bottom right) with respect to k and the corresponding parameter p for all variants (maximization, minimization, and all $\alpha \in [0, 1]$, see bottom left). Note that the split of the boxes is not proportional to the corresponding values of α. See Section 2 (paragraph "Graph parameter definitions.") for the definitions of the parameters. A line from a box for parameter p to a box *above* for parameter p' implies that $p \in \mathcal{O}(p')$ on all graphs. Thus, hardness results hold also for all parameters below and tractability results for all parameters above.

Known results. MaxPVC can be solved in $\mathcal{O}^*((\Delta+1)^k)$ time where Δ is the maximum degree of the input graph [30]. For the degeneracy d, Amini et al. [1] gave an $\mathcal{O}^*((dk)^k)$-time algorithm which was recently improved to an algorithm with running time $\mathcal{O}^*(2^{\mathcal{O}(dk)})$ [28]. Bonnet et al. [3] showed that in $\mathcal{O}^*(\Delta^k)$ time one can solve Max α-FCGP for all $\alpha > 1/3$ and Min α-FCGP for all $\alpha < 1/3$. Bonnet et al. [3] call these two problem cases *degrading*. This name reflects the fact that in Max α-FCGP with $\alpha > 1/3$, adding a vertex v to a set S increases the value at least as much as adding v to a superset of S.[2] This is because here one edge with both endpoints in S is less valuable than two edges each with one endpoint in S. In Min α-FCGP this effect is reversed since we aim to minimize val. The other problem cases are called *non-degrading*. For non-degrading problems, Bonnet et al. [3] achieved a running time of $\mathcal{O}^*((\Delta k)^{\mathcal{O}(k)})$ and asked whether they can also be solved in $\mathcal{O}^*(\Delta^{\mathcal{O}(k)})$ time. This question was answered positively by Shachnai and Zehavi [32], who showed that Max α-FCGP and Min α-FCGP can be solved in $\mathcal{O}^*(4^{k+o(k)}\Delta^k)$ time.

Kernelization has been studied only for special cases. Max $(k,n-k)$-Cut admits a polynomial problem kernel when parameterized by t [31]. This also gives a polynomial kernel for $k + \Delta$ since instances with $t > \Delta k$ are trivial no-instances. It is also known that Sparsest k-Subgraph admits a kernel with γk^2 vertices [23]. Here, γ is a parameter bounded by $\max(c, d+1)$ [16]. In contrast, Densest-k Subgraph is unlikely to admit a polynomial problem kernel when parameterized by $\Delta + k$ since Clique is a special case.

More broadly, for graph problems that are W[1]-hard for the standard parameter solution k size, the study of kernelization on sparse input graphs has received much attention in recent years [10, 11, 25, 29].

[2] Note that this matches the definition of submodularity.

Independent of our work, a polynomial *compression* for MaxPVC (the special case of Max α-FCGP with $\alpha = 1/2$) of size $(dk)^{\mathcal{O}(d)}$ was recently discovered by Panolan and Yaghoubizade [28].

Our results. We provide a complete picture of the parameterized complexity of Max α-FCGP and Min α-FCGP for all α with respect to the combination of k and five parameters describing the graph structure: the maximum degree Δ of G, the h-index of G, the degeneracy of G, the c-closure of G, and the vertex cover number vc of G. With the exception of the c-closure, all parameters are sparseness measures. The c-closure, first described by Fox et al. [16], measures how strongly a graph adheres to the triadic closure principle. Informally, the closure of a graph is small whenever all vertices with many common neighbors are also neighbors of each other. For a formal definition of all parameters refer to Section 2.

Our results are summarized by Figure 2. On a very general level, our main finding suggests that the degrading problems are much more amenable to FPT algorithms and kernelizations than their non-degrading counterparts. No such difference is observed when considering the running time of FPT algorithms for the parameter $k + \Delta$ but it becomes striking in the context of kernelization and when using secondary parameters that are smaller than Δ. Given the importance of the distinction between the degrading and non-degrading cases, we distinguish these subcases of Max α-FCGP and Min α-FCGP by name (Degrading Max α-FCGP, Non-Degrading Max α-FCGP, Degrading Min α-FCGP, Non-Degrading Min α-FCGP).

On a technical level, by introducing an annotated version of the problem that keeps track of removed vertices, we separate and unify arguments that deal with vertices identified as (not) being part of a solution. In particular, we show that by introducing vertex weights (called counter) we can deal with vertices whose contribution is substantially below or above the average contribution that is necessary to reach the threshold t. More precisely, if the contribution of a vertex v is much above t/k, then we can add v to the solution and if it is much below t/k, then we can remove v. As a consequence, we can show that the weights can be bounded in the maximum degree of the annotated instance. This gives the kernels for the parameter $k + \Delta$.

The main step in the more sophisticated kernelizations for the degeneracy d and the c-closure is now to decrease the maximum degree of the instance as this allows us to use the kernel for $k + \Delta$. To decrease the maximum degree, for these parameters, we make use of Ramsey bounds. More precisely, the Ramsey bounds help to find a large independent set I such that all vertices outside of I have only a bounded number of neighbors in I. This then allows to prove by pigeonholing the following for the vertex v of I with the currently worst contribution to the objective function: No matter what the optimal solution selects outside of I, there is always some vertex of $I \setminus \{v\}$ that gives at least as good a contribution to the final solution as v. For the parameter c, we also need an additional pigeonhole argument excluding large cliques in order to apply the Ramsey bound. For the parameter d, we establish a new constructive Ramsey bound for $K_{i,j}$-free graphs that may be of independent interest.

We remark that when we describe the kernel size for $\alpha > 0$ (for instance, Proposition 4.2), the factor $1/\alpha$ is hidden in the $\mathcal{O}$ notation. We would like to emphasize, however, that the exponents in the kernel size such as $\mathcal{O}(c)$ and $\mathcal{O}(d)$ do not depend on $1/\alpha$. On the other hand, the lower bounds such as Theorem 4.11 hold indeed for all α in the range corresponding to the case.

We believe that this general approach could be useful for other parameterizations that are not considered in this work. A somewhat surprising consequence of our kernelizations is that Partial Vertex Cover and Max $(k, n-k)$-Cut not only behave in the same way but that the kernels for both problems can also be obtained by the same algorithms.

Due to lack of space, several proofs (marked with $(\star)$) are deferred to the full version [22].

2 Preliminaries

For $q \in \mathbb{N}$, we write $[q]$ to denote the set $\{1, 2, \ldots, q\}$. For a graph G, we denote its *vertex set* by $V(G)$ and its *edge set* by $E(G)$. Let $X, Y \subseteq V(G)$ be vertex subsets. We use $G[X]$ to denote the *subgraph induced* by X. We let $G - X$ denote the graph obtained by *removing* the vertices in X. We denote by $N_G(X) \coloneqq \{y \in V(G) \setminus X \mid xy \in E(G), x \in X\}$ the *open neighborhood* and by $N_G[X] \coloneqq N_G(X) \cup X$ the *closed neighborhood* of X. By $E_G(X, Y) \coloneqq \{xy \in E(G) \mid x \in X, y \in Y\}$ we denote the set of edges *between* X and Y. As a shorthand, we set $E_G(X) \coloneqq E_G(X, X)$. Furthermore, we denote by $m_G(X, Y) \coloneqq |E_G(X, Y)|$ and $m_G(X) \coloneqq |E_G(X)|$ the *sizes* of these edge sets. For all these notations, when X is a singleton $\{x\}$ we may write x instead of $\{x\}$. Let $v \in V(G)$. We denote the *degree* of v by $\deg_G(v)$. We drop the subscript $\cdot_G$ when it is clear from context.

Graph parameter definitions. For more information on parameterized complexity, we refer to the standard monographs [9, 12]. We denote the size of a smallest vertex cover (a set of vertices that covers all edges) of a graph G by vc_G. The maximum and minimum degree of G are $\Delta_G \coloneqq \max_{v \in V(G)} \deg_G(v)$ and $\delta_G \coloneqq \min_{v \in V(G)} \deg_G(v)$, respectively. The degeneracy of G is $d_G \coloneqq \max_{S \subseteq V(G)} \delta_{G[S]}$. The *$h_G$-index* of a graph G is the largest integer h such that G has at least h vertices of degree at least h [14]. We say that G is c-closed for $c \coloneqq \max(\{0\} \cup \{|N_G(u) \cap N_G(v)| \mid uv \notin E(G)\}) + 1$ [16].

Ramsey numbers. Ramsey's theorem states that for every $p, q \in \mathbb{N}$, there exists an integer $R(p, q)$ such that any graph on at least $R(p, q)$ vertices contains either a clique of size p or an independent set of size q. The numbers $R(p, q)$ are referred to as *Ramsey numbers*. Although the precise values of Ramsey numbers are not known, some upper bounds have been proven. For instance, it holds that $R(p, q) \leq \binom{p+q-2}{p-1}$ (see e.g. [20]). The proof for this upper bound is constructive. More precisely, given a graph G on at least $\binom{p+q-2}{p-1}$ vertices, we can find in time $n^{\mathcal{O}(1)}$ either a clique of size p or an independent set of size q.

3 A Data Reduction Framework via Annotation

In this section, we introduce an annotated variant which allows easier handling for kernelization by giving more options for encoding information in the instances. Moreover, to avoid repeating certain basic arguments, we provide general data reduction rules and statements used in the subsequent sections. Finally, we describe how to reduce from the annotated to the non-annotated problem variants in polynomial time.

In the annotated problem variant, we have additionally as input a (possibly empty) partial solution $T \subseteq V(G)$ and $\mathsf{counter}\colon V \to \mathbb{N}$ which encodes for each vertex, the number of neighbors in the original graph that are guaranteed to be not in a solution. For a set $S \subseteq V(G)$, we set

- $\mathsf{counter}(S) \coloneqq \sum_{v \in S} \mathsf{counter}(v)$ and
- $\mathrm{val}_G(S) \coloneqq \alpha(m(S, V(G) \setminus S) + \mathsf{counter}(S)) + (1 - \alpha)m(S)$.

For $v \in S$ we set $\deg^{+c}(v) := \deg(v) + \mathsf{counter}(v)$.

Annotated Max α-FCGP
Input: A graph G, $T \subseteq V(G)$, $\mathsf{counter}\colon V(G) \to \mathbb{N}$, $k \in \mathbb{N}$, and $t \in \mathbb{Q}$.
Question: Is there a vertex set S, $T \subseteq S \subseteq V(G)$, of size k with $\mathrm{val}_G(S) \geq t$ (Max) or $\mathrm{val}_G(S) \leq t$ (Min), respectively?

For a partial solution $T \subseteq V(G)$, we define the *contribution* of a vertex v. Note that our definition slightly differs from that of Bonnet et al. [3]:

$$\begin{aligned}\mathrm{cont}(v, T) &:= \alpha \cdot (|N(v) \setminus T| + \mathsf{counter}(v)) + (1 - 2\alpha)|N(v) \cap T| \\ &= \alpha \deg^{+c}(v) + (1 - 3\alpha)|N(v) \cap T|\end{aligned}$$

This definition is chosen so that the value $\mathrm{val}(S)$ of a vertex set S computes as follows.

▶ **Lemma 3.1** ($\star$). *Let G be a graph and $S := \{v_1, \ldots, v_n\} \subseteq V(G)$ a vertex set. Then, it holds that* $\mathrm{val}(S) = \sum_{i \in [|S|]} \mathrm{cont}(v_i, \{v_1, \ldots, v_{i-1}\})$.

Main reduction rules. Annotations are helpful for data reductions in the following way: If we identify a vertex v that is (or is not) in a solution, then, we can simplify the instance as follows using the annotations.

▶ **Reduction Rule 3.2** (Inclusion Rule). If there is a solution S with $v \in S \setminus T$, then add v to T. If there is a vertex $v \in T$ with $\mathsf{counter}(v) > 0$, then reduce t by $\alpha \cdot \mathsf{counter}(v)$ and set $\mathsf{counter}(v) := 0$.

▶ **Reduction Rule 3.3** (Exclusion Rule). If there is a solution S with $v \notin S$, then for each $u \in N(v)$ increase $\mathsf{counter}(u)$ by one and remove v from G.

The reduction rules themselves are simple. The difficulty lies in identifying vertices that are included in or excluded from some solution. In the respective arguments, we use the subsequently discussed notion of better vertices.

Better vertices. The following notion captures a situation that frequently appears in our arguments and allows for simple exchange arguments (see following lemma).

▶ **Definition 3.4.** *A vertex $v \in V(G)$ is* better *than $u \in V(G)$ with respect to a vertex set $T \subseteq V(G)$ if* $\mathrm{cont}(v, T) \geq \mathrm{cont}(u, T)$ *for the maximization variant (if* $\mathrm{cont}(v, T) \leq \mathrm{cont}(u, T)$ *for the minimization variant).*

A vertex $v \in V(G)$ is strictly better *than $u \in V(G)$ if for all $T \subseteq V(G)$ of size at most k we have* $\mathrm{cont}(v, T) \geq \mathrm{cont}(u, T)$ *for the maximization variant (*$\mathrm{cont}(v, T) \leq \mathrm{cont}(u, T)$ *for the minimization variant).*

When we simply say that v is better than u, we mean that v is better than u with respect to the empty set. The following lemma immediately follows from Lemma 3.1.

▶ **Lemma 3.5.** *Let S be a solution of an instance of* α-FCGP. *Suppose that there are two vertices $v \in S$ and $v' \notin S$ such that v' is better than v with respect to $S \setminus \{v\}$ or v' is strictly better than v. Then, $S' := (S \setminus \{v\}) \cup \{v'\}$ is also a solution.*

Proof. We give a proof for the maximization variant; the minimization variant follows analogously. By Lemma 3.1, we have $\mathrm{val}(S') = \mathrm{val}(S \setminus \{v\}) + \mathrm{cont}(v', S \setminus \{v\}) \geq \mathrm{val}(S \setminus \{v\}) + \mathrm{cont}(v, S \setminus \{v\}) = \mathrm{val}(S)$. ◀

Observe that the contribution of any vertex v differs from $\alpha \deg^{+c}(v)$ by at most $|(1-3\alpha)k|$. This observation allows us to identify some strictly better vertices in the following. This is helpful when we wish to apply the second part of Lemma 3.5 on strictly better vertices.

▶ **Lemma 3.6** (⋆). *Let $u, v \in V(G)$. Vertex v is strictly better than u if*

- *(Maximization:)* $\alpha \deg^{+c}(u) \leq \alpha \deg^{+c}(v) - |(1-3\alpha)k|$.
- *(Minimization:)* $\alpha \deg^{+c}(u) \geq \alpha \deg^{+c}(v) + |(1-3\alpha)k|$.

Reduction to the non-annotated problem. The following two lemmas (for maximization and minimization variant respectively) remove annotations and generate an equivalent instance of α-FCGP. We remark that the resulting instance size depends on $\Gamma := \max_{v \in V(G)} \mathsf{counter}(v) + 1$. We obtain an upper bound on Γ in terms of $k + \Delta$ in the next section.

▶ **Lemma 3.7.** *Given an instance $\mathcal{I} := (G, T, \mathsf{counter}, k, t)$ of* Annotated Max α-FCGP *with $\alpha \in (0, 1]$, we can compute an equivalent instance $\mathcal{I}'$ of* Max α-FCGP *of size $\mathcal{O}((\Delta + \Gamma) \cdot |V(G)| + k \cdot |T|)$ in polynomial time.*

Proof. We may assume that G has at least k vertices (otherwise the lemma holds for a trivial No-instance $\mathcal{I}'$). We construct an equivalent instance $\mathcal{I}' := (G', k, t')$ of Max α-FCGP. The graph G' is obtained from G as follows:

1. Add $\mathsf{counter}(v) + \lfloor 1/\alpha \rfloor$ degree-one neighbors to every vertex $v \in V(G)$.
2. Additionally, add $\ell := \Delta + \Gamma + |1/\alpha - 3| \cdot k + \lfloor 1/\alpha \rfloor$ degree-one neighbors to every vertex $v \in T$.

By L_v we denote the set of degree-one vertices added to vertex $v \in V(G)$ and by $L := \bigcup_{v \in V(G)} L_v$ we denote the set of all newly added leaf vertices. To conclude the construction of $\mathcal{I}'$, we set $t' := t + \alpha(\ell \cdot |T| + \lfloor 1/\alpha \rfloor \cdot k)$. Since G has at most $\Delta \cdot |V(G)|$ edges and we add at most $\mathcal{O}((\Gamma + 1) \cdot |V(G)| + (\Delta + k) \cdot |T|)$ edges, we see that G' has at most $\mathcal{O}((\Delta + \Gamma) \cdot |V(G)| + k \cdot |T|)$ edges.

Next, we prove the equivalence between $\mathcal{I}$ and $\mathcal{I}'$. For a solution S of $\mathcal{I}$, its value in G' is increased by $\alpha \cdot \lfloor 1/\alpha \rfloor$ for every vertex in S and additionally, by $\alpha \cdot \ell$ for every vertex in T, amounting to $t + \alpha(\ell \cdot |T| + \lfloor 1/\alpha \rfloor \cdot k)$.

Conversely, consider a solution S' of $\mathcal{I}$. First, we show that there is a solution containing all vertices of T and no leaf vertex of L using Lemma 3.5. Suppose that for some vertex $v \in V(G)$, one of its degree-one neighbors $v' \in L_v$ is in S' but not v itself. We then have $\mathrm{cont}(v', S' \setminus \{v\}) = \alpha$ and $\mathrm{cont}(v, S' \setminus \{v\}) \geq \alpha$, implying that $(S' \setminus \{v'\}) \cup \{v\}$ is also a solution by Lemma 3.5. Thus, in the following we can assume that $S' \cap L_v = \emptyset$ for every vertex $v \in V(G) \setminus S'$. If there is a vertex $v' \in S' \cap L_v$ for some $v \in V(G)$, then by the assumption that $|V(G)| \geq k$, the pigeonhole principle gives us a vertex $w \in V(G) \setminus S'$ with $S' \cap L_w = \emptyset$. Since $|L_w| \geq \mathsf{counter}(w) + \lfloor 1/\alpha \rfloor \geq \lfloor 1/\alpha \rfloor$, we have $\mathrm{cont}(w, S' \setminus \{v'\}) \geq \alpha \cdot \lfloor 1/\alpha \rfloor \geq \alpha(1/\alpha - 1) = 1 - \alpha$. We thus have $\mathrm{cont}(v', S \setminus \{v'\}) = 1 - \alpha \leq \mathrm{cont}(w, S' \setminus \{v'\})$. Hence, $(S' \setminus \{v'\}) \cup \{w\}$ is a solution, again by Lemma 3.5. Thus, in the following, we can assume that $S' \cap L = \emptyset$.

So we may assume that S' consists only of vertices in G. Suppose that some vertex $v \in T$ is not in S'. For any vertex $v' \in S' \setminus T$, we have $\deg(v') \leq \deg_G(v) + \mathsf{counter}(v) + \lfloor 1/\alpha \rfloor \leq \Delta + \Gamma + \lfloor 1/\alpha \rfloor$. So we have $\deg(v') \geq \Delta + \Gamma + |1/\alpha| \cdot k + \lfloor 1/\alpha \rfloor \geq \deg(v) + |1/\alpha - 3| \cdot k$. Applying Lemma 3.6 with $\mathsf{counter}(v) = \mathsf{counter}(v') = 0$, we obtain that v is strictly better than v'. Now, it follows from Lemma 3.5 that $\mathcal{I}'$ has a solution S' such that $T \subseteq S' \subseteq V(G')$. Hence, S' is also a solution for $\mathcal{I}$. ◀

▶ **Lemma 3.8** (⋆). *Given an instance $\mathcal{I} := (G, T, \mathsf{counter}, k, t)$ of* Annotated Min α-FCGP *for $\alpha \in (0, 1]$, we can compute an equivalent instance $\mathcal{I}'$ of* Min α-FCGP *of size $\mathcal{O}((\Delta + \Gamma)^3 \cdot |V(G)|)$ in polynomial time.*

4 Parameterization By Maximum Degree

4.1 Polynomial Kernels in Degrading Cases

Recall that in the degrading cases we have $\alpha \in (1/3, 1]$ for maximization and $\alpha \in [0, 1/3)$ for minimization. Furthermore, recall that for two vertices u and v, v is said to be better than u if $\mathrm{cont}(v, T) \geq \mathrm{cont}(u, T)$ (vice versa for the minimization variant).

For the annotated version we define $\Delta_{\overline{T}} := \max_{v \in V(G) \setminus T} \deg(v)$. Clearly, $\Delta_{\overline{T}} \leq \Delta$.

▶ **Reduction Rule 4.1** ($\star$)**.** Let $\mathcal{I}$ be an instance of Annotated Degrading α-FCGP. If there are more than $\Delta_{\overline{T}} k + 1$ vertices that are better than v with respect to T, then apply the Exclusion Rule (Reduction Rule 3.3) to v.

Next, we show that the exhaustive application of Reduction Rule 4.1 yields a polynomial kernel for Degrading α-FCGP.

▶ **Proposition 4.2.** Degrading α-FCGP *has a kernel of size*

- $\mathcal{O}(\Delta^2 k)$ *for maximization and* $\alpha \in (1/3, 1]$*, and*
- $\mathcal{O}(\Delta^4 k)$ *for minimization and* $\alpha \in (0, 1/3)$*.*

Proof. Given an instance of Degrading α-FCGP, we transform it into an equivalent instance of Annotated Degrading α-FCGP and apply Reduction Rule 4.1 exhaustively. Observe that $|V(G)| \leq \Delta_{\overline{T}} k + 1 \leq \Delta k + 1$. Moreover, we have $T = \emptyset$ and $\Gamma \leq \Delta$ since each neighbor of a vertex can increase its counter by at most one. By Lemma 3.7 (maximization) resp. Lemma 3.8 (minimization), we obtain an equivalent instance of α-FCGP of size $\mathcal{O}(\Delta^2 k)$ (maximization) resp. $\mathcal{O}(\Delta^4 k)$ (minimization). ◀

Note that Proposition 4.2 does not cover the case $\alpha = 0$ for minimization. Note that this problem is referred to as Sparsest k-Subgraph. We remark that this is complemented by Proposition 6.4, in which we provide a kernel of size $\mathcal{O}(d^2 k)$. Since $d \leq \Delta$, this implies also a kernel of size $\mathcal{O}(\Delta^2 k)$.

Proposition 4.2 basically shows that given an instance of Degrading α-FCGP, we can find in polynomial time an equivalent instance of Degrading α-FCGP of size $\mathcal{O}(\Delta + k)^{O(1)}$. In the following, we will show that an equivalent instance of Degrading α-FCGP that has size $(\Delta + k)^{\mathcal{O}(1)}$ can be constructed even if an instance of Annotated Degrading α-FCGP is given (see Proposition 4.10). Proposition 4.10 plays an important role in kernelizations in subsequent sections. Essentially, the task of kernelization boils down to bounding the maximum degree Δ by Proposition 4.10.

As shown in the proof of Proposition 4.2, the instance size becomes polynomial in $k + \Delta$ by exhaustively applying Reduction Rule 4.1. Recall that in Section 3, we presented a polynomial-time procedure to remove annotations with an additional polynomial factor in $\Delta + \Gamma$ on the instance size, where Γ denotes the maximum counter. To prove Proposition 4.10, it remains to bound Γ for Annotated Degrading α-FCGP.

Bounding the largest counter Γ. Throughout the section, let $k' := k - |T|$ and $t' := t - \mathrm{val}(T)$. First, we identify vertices which are contained in a solution, if one exists.

▶ **Definition 4.3.** *Let $\mathcal{I}$ be a Yes-instance of* Annotated Degrading α-FCGP*. A vertex $v \in V(G) \setminus T$ is called* satisfactory *if*

- *(Maximization:)* $\mathrm{cont}(v, T) \geq t'/k' + (3\alpha - 1)(k - 1)$ *and* $\alpha \in (1/3, 1]$*.*
- *(Minimization:)* $\mathrm{cont}(v, T) \leq t'/k' - (1 - 3\alpha)(k - 1)$ *and* $\alpha \in (0, 1/3)$*.*

▶ **Reduction Rule 4.4** ($\star$). Let $\mathcal{I}$ be an instance of Annotated Degrading α-FCGP with $\alpha > 0$ and let $v \in V(G) \setminus T$ be a satisfactory vertex. Apply the Inclusion Rule (Reduction Rule 3.2) on vertex v.

We henceforth assume that Reduction Rule 4.4 is exhaustively applied on every satisfactory vertex. Next, we identify vertices which are not contained in any solution.

▶ **Definition 4.5.** *Let $\mathcal{I}$ be a Yes-instance of* Annotated Degrading α-FCGP*. A vertex $v \in V(G) \setminus T$ is called* needless *if*

- *(Maximization:)* $\text{cont}(v, T) \le t'/k' - (3\alpha - 1)(k-1)^2$ *for maximization and* $\alpha \in (1/3, 1]$.
- *(Minimization:)* $\text{cont}(v, T) \ge t'/k' + (1 - 3\alpha)(k-1)^2$ *for minimization and* $\alpha \in (0, 1/3)$.

▶ **Reduction Rule 4.6** ($\star$). Let $\mathcal{I}$ be an instance of Annotated Degrading α-FCGP with $\alpha > 0$ and let $v \in V(G) \setminus T$ be a needless vertex. Apply the Exclusion Rule (Reduction Rule 3.3) on vertex v.

We henceforth assume that Reduction Rule 4.6 is applied on every needless vertex. The following reduction rule decreases the counter of each vertex in $V(G) \setminus T$. After this rule is exhaustively applied, we may assume that $\mathsf{counter}(v) = 0$ for at least one vertex $v \in V(G) \setminus T$. Recall that we already have $\mathsf{counter}(v) = 0$ for every vertex in T.

▶ **Reduction Rule 4.7.** If $\mathsf{counter}(v) > 0$ for every vertex $v \in V(G) \setminus T$, then decrease $\mathsf{counter}(v)$ by 1 for every vertex $v \in V(G) \setminus T$ and decrease t by $\alpha k'$.

Next, we show that after the exhaustive application of Reduction Rule 4.7 the counter of each vertex is bounded polynomially in terms of Δ and k.

▶ **Lemma 4.8.** *Let $\mathcal{I}$ be a Yes-instance of* Annotated Degrading α-FCGP *with $\alpha > 0$. We have $\mathsf{counter}(v) \in \mathcal{O}(\Delta + k^2)$ for every vertex $v \in V(G) \setminus T$.*

Proof. First and foremost, observe that there exists at least one vertex $u \in V(G) \setminus T$ with $\mathsf{counter}(u) = 0$, since otherwise Reduction Rule 4.7 is still applicable. Since Reduction Rule 3.3 is applied to each needless vertex, we conclude that every vertex in $V(G) \setminus T$ has contribution at least $t'/k' - (3\alpha - 1)(k-1)^2$ for maximization. Furthermore, since Reduction Rule 3.2 is applied to each satisfactory vertex, we conclude that every vertex $v \in V(G) \setminus T$ has contribution at least $t'/k' - (1 - 3\alpha)(k-1)$ for minimization. In particular, we have

$$\text{cont}(u, T) \ge t'/k' - (3\alpha - 1)(k-1)^2 \text{ for maximization, and}$$
$$\text{cont}(u, T) \ge t'/k' - (1 - 3\alpha)(k-1) \text{ for minimization.}$$

Since also $\text{cont}(u, T) = \alpha \cdot \deg(u) + (1 - 3\alpha)|N(u) \cap T|$ we obtain that

$$t'/k' \le \alpha \cdot \deg(u) + (1 - 3\alpha)|N(u) \cap T| + (3\alpha - 1)(k-1)^2 \text{ for maximization, and} \quad (1)$$
$$t'/k' \ge \alpha \cdot \deg(u) + (1 - 3\alpha)[(k-1) + |N(u) \cap T|] \text{ for minimization.} \quad (2)$$

Moreover, since Reduction Rule 3.2 is applied to each satisfactory vertex, we conclude that every vertex $v \in V(G) \setminus T$ has contribution at most $t'/k' + (3\alpha - 1)(k-1)$ for maximization. Furthermore, since Reduction Rule 3.3 is applied to each needless vertex, we conclude that every vertex in $V(G) \setminus T$ has contribution at most $t'/k' + (1 - 3\alpha)(k-1)^2$ for minimization. This implies that in particular

$$\alpha \cdot \mathsf{counter}(v) \le t'/k' + (3\alpha - 1)(k-1) \text{ for maximization, and.} \quad (3)$$
$$\alpha \cdot \mathsf{counter}(v) \le t'/k' + (1 - 3\alpha)(k-1)^2 \text{ for minimization.} \quad (4)$$

For maximization and $\alpha \in (1/3, 1]$ it then follows from Equations (1) and (3) that

$$\mathsf{counter}(v) \leq \deg(u) + \frac{3\alpha - 1}{\alpha}[k(k-1) - |N(v) \cap T|] \in \mathcal{O}(\Delta + k^2).$$

For minimization and $\alpha \in (0, 1/3)$ it then follows from Equations (2) and (4) that

$$\mathsf{counter}(v) \leq \deg(u) + \frac{1 - 3\alpha}{\alpha}[k(k-1) + |N(v) \cap T|] \in \mathcal{O}(\Delta + k^2) \text{ for minimization.}$$

This concludes the proof. ◀

Putting everything together. We use the following proposition in our kernels for Degrading α-FCGP. Therefore, we first transform the instance into an equivalent instance of Annotated Degrading α-FCGP. Second, we apply our reduction rules. Third, we reduce back to the unannotated version.

▶ **Proposition 4.9** (⋆). *Let $\alpha > 0$. Given an instance $(G, T, \mathsf{counter}, k, t)$ of*

- Annotated Degrading Max α-FCGP*, we can compute in polynomial time an equivalent* Degrading Max α-FCGP *instance of size $\mathcal{O}(|V(G)|^2 + |V(G)|k^2) \subseteq \mathcal{O}(|V(G)|^3)$, and*
- Annotated Degrading Min α-FCGP*, we can compute in polynomial time an equivalent* Degrading Min α-FCGP *instance of size $\mathcal{O}(|V(G)| \cdot (|V(G)| + k^2)^3) \subseteq \mathcal{O}(|V(G)|^7)$.*

▶ **Proposition 4.10.** *Given an instance $(G, T, \mathsf{counter}, k, t)$ of* Annotated Degrading α-FCGP *with $\alpha > 0$, we can compute in polynomial time an equivalent* Degrading α-FCGP *instance of size $\mathcal{O}((\Delta + k)^{\mathcal{O}(1)})$.*

Proof. Follows from Lemmas 3.7–3.8, and Proposition 4.2. ◀

4.2 No Polynomial Kernels in Non-Degrading Cases

Note that if $\alpha = 0$, then Max α-FCGP corresponds to Densest k-Subgraph. It is already known that Densest k-Subgraph does not admit a polynomial kernel for $\Delta + k$ [26]. We strengthen and generalize this result: First, we observe that Densest k-Subgraph does not admit a polynomial kernel for k, even when $\Delta = 3$. Second, we extend this negative result to Non-Degrading Max α-FCGP and Non-Degrading Min α-FCGP when Δ is a constant.

▶ **Theorem 4.11** (⋆). *Unless* coNP ⊆ NP/poly*,*

1. Non-Degrading Max α-FCGP *on subcubic graphs does not admit a polynomial kernel for k, and*
2. Non-Degrading Min α-FCGP *on graphs with constant maximum degree does not admit a polynomial kernel for k.*

5 Parameterization by c-closure

Hardness for the non-degrading case. We start with showing that the Non-Degrading case is intractable.

▶ **Theorem 5.1** (⋆). Non-Degrading Max α-FCGP *remains W[1]-hard with respect to the solution size k even on 2-closed and 2-degenerate graphs.*

▶ **Theorem 5.2** (⋆). Non-Degrading Min α-FCGP *remains W[1]-hard with respect to the solution size k even on 2-closed graphs.*

A $k^{\mathcal{O}(c)}$-size kernel for the degrading case. In contrast to the non-degrading case, we develop a kernel of size $k^{\mathcal{O}(c)}$ for the degrading case. To this end, we apply a series of reduction rules to obtain an upper bound of $k^{\mathcal{O}(c)}$ on the maximum degree. Then, the kernel of size $k^{\mathcal{O}(c)}$ follows from Proposition 4.10. In order to upper-bound the maximum degree, we rely on a polynomial Ramsey bound for c-closed graphs [24].

▶ **Lemma 5.3** ([24, Lemma 3.1]). *Any c-closed graph G on at least $R_c(a,b) := (c-1)\cdot\binom{b-1}{2} + (a-1)(b-1)+1$ vertices contains a clique of size a or an independent set of size b. Moreover, a clique of size a or an independent set of size b can be found in polynomial time.*

Using a similar approach as Reduction Rule 4.1 (but exploiting the c-closure instead of the maximum degree) yields the following.

▶ **Reduction Rule 5.4.** Let $\mathcal{I}$ be an instance of ANNOTATED DEGRADING α-FCGP. Let $v \in V(G)$ be some vertex and let $X_v \subseteq N(v)$ be the set of vertices better than v. If $|X_v| > (c-1)k$, then apply the Exclusion Rule (Reduction Rule 3.3) to v.

▶ **Lemma 5.5.** *Reduction Rule 5.4 is correct.*

Proof. We provide a proof for the maximization version; the minimization version follows analogously. Let S be a solution. Assume that $v \in S$ (we are done otherwise). We show that there is a vertex $v' \neq v$ such that $S' := (S \setminus \{v\}) \cup \{v'\}$ constitutes a solution. By Lemma 3.5, it suffices to show that $\operatorname{cont}(v', S \setminus \{v\}) \geq \operatorname{cont}(v, S \setminus \{v\})$. Let $S'_v := S \setminus N[v]$. Each vertex in S'_v is, by definition, nonadjacent to v, and hence it shares at most $c-1$ neighbors in common with v. This implies $|X_v \setminus N(S'_v)| \geq |X_v| - (c-1)\cdot|S'_v| > 0$ as $X_v \subseteq N(v)$. Thus, there exists a vertex $v' \in X_v \setminus N(S'_v)$, that is, $N(v') \cap S'_v = \emptyset$. Then, we have $N(v') \cap (S \setminus \{v\}) \subseteq S \cap N(v)$ and thus $|N(v') \cap (S \setminus \{v\})| \leq |N(v) \cap (S \setminus \{v\})|$. Moreover, we have $\alpha \deg^{+c}(v') \geq \alpha \deg^{+c}(v)$ (recall that v' is better than v). Since $\alpha \in (1/3, 1]$, it follows that

$$\begin{aligned}&\operatorname{cont}(v', S \setminus \{v\}) = \alpha \deg^{+c}(v') + (1-3\alpha)|N(v') \cap (S \setminus \{v\})|\\ &\geq \alpha \deg^{+c}(v) + (1-3\alpha)|N(v) \cap (S \setminus \{v\})| = \operatorname{cont}(v, S \setminus \{v\}).\end{aligned}$$

This concludes the proof. ◀

Note that if there is a clique of size $(c-1)k+1$, then Reduction Rule 5.4 applies to one of the vertices with the smallest contribution. Thus, applying Reduction Rule 5.4 exhaustively removes all cliques of size $(c-1)k+1$.

▶ **Lemma 5.6.** *Let $v \in V(G)$ be a vertex such that $\deg(v) \geq R_c((c-1)k+1, (k+1)k^{c-2})$. Then, we can find in polynomial time a set X of $i \in [c-1]$ vertices and an independent set I with the following properties:*

(i) *The set X contains v.*

(ii) *The set $I \subseteq \bigcap_{x \in X} N(x)$ is an independent set of size at least $(k+1)k^{c-i}$.*

(iii) *For every vertex $u \in V(G) \setminus X$, it holds that $|N(u) \cap I| \leq (k+1)k^{c-i-1}$.*

Proof. Since there is no clique of size $(c-1)k+1$, there is an independent set I_v of size $(k+1)k^{c-2}$ in $N(v)$ by Lemma 5.3 (which can be found in polynomial time). Let X be an inclusion-wise maximal set of i vertices including v such that $|\bigcap_{x \in X} N(x) \cap I_v| > (k+1)k^{c-i}$. One of such sets can be found by the following polynomial-time algorithm: We start with $X := \{v\}$ and $i := 1$. We will maintain the invariant that $|X| = i$. If there exists a vertex $v' \in V(G) \setminus X$ with $|N(v') \cap \bigcap_{x \in X} N(x) \cap I_v| > (k+1)k^{c-i-1}$, then we add v' to X and increase i by 1. We keep doing so until there remains no such vertex v'.

We show that this algorithm terminates for $i = |X| \leq c - 1$. Assume to the contrary that the algorithm continues for $i = c - 1$. We then have that $|N(v') \cap \bigcap_{x \in X} N(x) \cap I_v| > (k + 1)k^{c-i-1} = k + 1 \geq 2$ for some vertex $v' \in V(G) \setminus X$. Since I_v is an independent set, the set $N(v') \cap \bigcap_{x \in X} N(x) \cap I_v$ contains two nonadjacent vertices. Note, however, that these two vertices have at least $|X \cup \{v\}| = c$ common neighbors, contradicting the c-closure of G.

Finally, we show that a set X found by this algorithm and $I := \bigcap_{x \in X} N(x) \cap I_v$ satisfy the three properties of the lemma. We have $|\bigcap_{x \in X} N(x) \cap I_v| = |N(v') \cap \bigcap_{x \in X \setminus \{v\}} N(x) \cap I_v| > (k + 1)k^{c-(i-1)-1} = (k + 1)k^{c-i}$, where v' is the last vertex added to X. Moreover, since X is inclusion-wise maximal, we have $|N(u) \cap I| = |N(u) \cap \bigcap_{x \in X} N(x) \cap I_v| \leq (k + 1)k^{c-i-1}$ for every vertex $u \in V(G) \setminus X$. ◀

▶ **Reduction Rule 5.7.** Let $\mathcal{I}$ be an instance of ANNOTATED DEGRADING α-FCGP. Let X, I be as specified in Lemma 5.6 and let $v \in I$ be a vertex such that every other vertex in I is better than v. If $k \geq 2$, then apply the Exclusion Rule (Reduction Rule 3.3) to v.

▶ **Lemma 5.8.** *Reduction Rule 5.7 is correct.*

Proof. Again, we show the proof for the maximization variant; the minimization variant follows analogously. For the sake of contradiction, assume that every solution S contains v. By Lemma 5.6, every vertex $u \in V(G) \setminus X$ has at most $(k + 1)^{c-i}$ neighbors in I. Moreover, since I is an independent set, we have $|I \cap N[v']| = 1$ for every vertex $v' \in I$ (including v). For $S' := S \setminus X$, we have

$$\begin{aligned}|I \setminus N[S']| &\geq |I| - |I \cap N[v]| - |I \cap N[S' \setminus \{v\}]| \\ &\geq (k + 1)k^{c-i} - (k - 1)(k + 1)k^{c-i-1} - 1 = k^{c-i} + k^{c-i-1} - 1 > 0.\end{aligned}$$

Let v' be an arbitrary vertex in $I \setminus N[S']$. We show that $\text{cont}(v', S \setminus \{v\}) \geq \text{cont}(v, S \setminus \{v\})$. By Lemma 3.5, this would imply that $(S \setminus \{v\}) \cup \{v'\}$ is a solution not containing v. Since v and v' are both adjacent to all vertices of X and $\alpha \in (1/3, 1]$, we have $|N(v) \cap (S \setminus \{v\})| > |X \cap (S \setminus \{v\})|$. We thus have

$$\begin{aligned}\text{cont}(v', S \setminus \{v\}) &= \alpha \deg^{+c}(v') + (1 - 3\alpha)|X \cap (S \setminus \{v\})| \\ &\geq \alpha \deg^{+c}(v) + (1 - 3\alpha)|X \cap (S \setminus \{v\})| \\ &\geq \alpha \deg^{+c}(v) + (1 - 3\alpha)|N(v) \cap (S \setminus \{v\})| = \text{cont}(v, S \setminus \{v\}).\end{aligned}$$

Here, the first inequality follows from the fact that v' is better than v. ◀

By applying these reduction rules, we can ensure that $\Delta \leq R_c((c-1)k+1, (k+1)k^{c-2}) \in k^{\mathcal{O}(c)}$. Proposition 4.10 leads to the following:

▶ **Proposition 5.9.** DEGRADING α-FCGP *has a kernel of size* $k^{\mathcal{O}(c)}$.

Matching lower bounds. Next, we show that the kernels provided in Theorem 5.9 cannot be improved under standard assumptions.

▶ **Proposition 5.10** ($\star$). DEGRADING MAX α-FCGP *has no kernel of size* $\mathcal{O}(k^{c-3-\epsilon})$ *unless* coNP $\subseteq$ NP/poly.

▶ **Proposition 5.11** ($\star$). *For each* $\alpha \in (0, 1/3)$, MIN α-FCGP *does not admit a kernel of size* $\mathcal{O}(k^{c-3-\epsilon})$ *unless* coNP $\subseteq$ NP/poly.

Note that Min α-FCGP for $\alpha = 0$ is equivalent to Sparsest k-Subgraph which admits a kernel of size $\mathcal{O}(c^2k^3)$ [23].

Now, Propositions 5.9–5.11 imply the following.

▶ **Theorem 5.12.** Degrading α-FCGP *admits a kernel of size* $k^{\mathcal{O}(c)}$*. For* $\alpha > 0$*,* Degrading α-FCGP *does not admit a kernel of size* $k^{o(c)}$ *unless* coNP $\subseteq$ NP/poly*.*

6 Parameterization by Degeneracy

Minimization variant. We start with the minimization variant which turns out to be easier than the maximization variant. This is most likely because of the following bound on t.

▶ **Lemma 6.1** ($\star$)**.** *In non-trivial instances* (G, k, t) *of* Min α-FCGP *we have* $t \leq dk$*.*

Shachnai and Zehavi [32] showed that Min α-FCGP with $\alpha \in (0, 1]$ admits an FPT-algorithm with respect to $k + t$. Hence, we obtain the following.

▶ **Corollary 6.2.** Min α-FCGP *for* $\alpha > 0$ *is FPT parameterized by* $d + k$*.*

Naturally, we may now ask whether this FPT result can be strengthened to a polynomial kernel. As shown by Theorem 4.11, the non-degrading case of Min α-FCGP does not admit a polynomial kernel even on graphs with constant maximum degree which implies constant degeneracy. In contrast, the degrading has a kernel whose size is polynomial in $d + k$.

▶ **Theorem 6.3** ($\star$)**.** Degrading Min α-FCGP *admits a kernel of size* $(d + k)^{\mathcal{O}(1)}$*.*

▶ **Proposition 6.4** ($\star$)**.** Sparsest k-Subgraph *admits a kernel with* $\mathcal{O}(dk)$ *vertices and of size* $\mathcal{O}(d^2k)$*.*

Maximization variant. Recall, that MaxPVC is the special case of Max α-FCGP with $\alpha = 1/2$. Amini et al. [1] showed that MaxPVC can be solved in $\mathcal{O}^*((dk)^k)$ time. Adapting this algorithm leads to an FPT-algorithm for α-FCGP with respect to $d + k$ for $\alpha \neq 0$:

▶ **Proposition 6.5** ($\star$)**.** Degrading α-FCGP *can be solved in* $\mathcal{O}^*((dk)^k)$ *time for* $\alpha \neq 0$*.*

The rest of this section is devoted to the proof of the next theorem.

▶ **Theorem 6.6.** Degrading Max α-FCGP *admits a kernel of size* $k^{\mathcal{O}(d)}$ *but, unless* coNP $\subseteq$ NP/poly*, no kernel of size* $\mathcal{O}(k^{d-2-\epsilon})$*.*

In particular, this implies that MaxPVC admits a kernel of size $k^{\mathcal{O}(d)}$. We remark that a compression of size $(dk)^{\mathcal{O}(d)}$ was obtained independently by Panolan and Yaghoubizade [28].

A kernel for biclique-free graphs in the degrading case. We next develop a kernel of size $k^{\mathcal{O}(d)}$. In fact, our algorithm works for *biclique-free graphs* – graphs that do not have a biclique $K_{a,b}$ as a subgraph for $a \leq b \in \mathbb{N}$. Note that a d-degenerate graph has no $K_{d+1,d+1}$ as a subgraph, since otherwise every vertex in $K_{d+1,d+1}$ has at least degree $d + 1$.

Note that a clique of size $a + b$ contains $K_{a,b}$ as a subgraph. So given a graph G with no occurrence of $K_{a,b}$ on at least $\binom{a+b+k-2}{k-1} \in k^{\mathcal{O}(a+b)}$ vertices, one can find an independent set of size k in polynomial time (see Section 2). We show that this upper bound on the number of vertices can be improved: the sum $a + b$ in the exponent can be replaced by $\min\{a, b\}$.

▶ **Lemma 6.7.** *For* $a \leq b \in \mathbb{N}$*, let* G *be a graph that contains no* $K_{a,b}$ *as a subgraph. If* G *has at least* $R(k)$ *vertices, then we can find in polynomial time an independent set of size* k*, where* $R(k) \in (a + b)^{\mathcal{O}(a)} \cdot k^a$*.*

Proof. We first show that if G has at least $k + b\binom{k}{a} + \sum_{\ell \in [a-1]} R(a+b, \ell+1)\binom{k}{\ell}$ vertices, then it contains an independent set of size k. We give an algorithm to find an independent set of size k in polynomial time later. Let I be a maximum independent set in G. We assume for contradiction that $|I| < k$. We prove that there are at most $t\binom{k}{a}$ vertices that have at least a neighbors in I and that there are at most $\sum_{\ell \in [a-1]} R(a+b, \ell+1)$ vertices that have at most $a-1$ neighbors in I.

For each subset $X \subseteq I$ of size exactly a, note that there are at most b vertices v such that $N(v) \supseteq X$, since otherwise there is a $K_{a,b}$ in G. It follows that the number of vertices with at least a neighbors in I is at most $t\binom{|I|}{a} \le b\binom{k}{a}$. Consider a set $X \subseteq I$ of size $\ell \in [a-1]$. Let $V_X \coloneqq \{v \in V(G) \setminus I \mid N(v) \cap I = X\}$. Then, there is no independent set I' of size $\ell + 1$ in V_X, since otherwise $(I \setminus X) \cup I'$ is an independent set of size at least $|I| + 1$, contradicting the fact that I is an independent set of maximum size. Moreover, there is no clique of size $a+b$ in V_X. Thus, $|V_X| < R(a+b, \ell+1)$. The number of vertices with at most $a-1$ neighbors in I is then at most $\sum_{X \subseteq I, |X| = \ell \in [a-1]} R(a+b, \ell+1)\binom{|I|}{\ell} \le \sum_{\ell \in [a-1]} R(a+b, \ell+1)\binom{k}{\ell}$.

We turn the argument above into a polynomial-time algorithm as follows. Suppose that we have an independent set I' of size smaller than k. As discussed above, there are at most $b \cdot \binom{k}{a}$ vertices that have at least s neighbors in I'. Hence, there is a vertex set $X \subseteq I'$ of size ℓ such that $|V_X| > R(a+b, \ell+1)$. Note that X can be found in polynomial time, for instance, by counting the number of vertices v' such that $N(v') \cap I' = N(v) \cap I'$ for each vertex $v \in V(G)$. We can then find an independent set I'' of size $\ell + 1$ in X (this can be done in polynomial time as discussed in Section 2). This way, we end up with an independent set $(I' \setminus X) \cup I''$ of size at least $|I'| + 1$. Note that this procedure of finding an independent set of greater size is repeated at most k times, and thus the overall running time is polynomial. ◀

We remark that for fixed $a \le b \in \mathbb{N}$, Lemma 6.7 gives us an $\mathcal{O}(n^{1-1/a})$-approximation algorithm for Independent Set that runs in n^{ℓ} time with a constant ℓ not depending on a or b. An $\mathcal{O}(n^{1-1/a})$-approximation algorithm is known on graphs where $K_{a,b}$ is excluded as an *induced subgraph* [4, 13]. However, these algorithms have running time $n^{\Omega(a)}$.

We then apply Lemma 6.7 to obtain a lemma analogous to Lemma 5.6.

► **Lemma 6.8.** *Let $v \in V(G)$ be a vertex such that $\deg(v) \ge R(bk^{a-1})$. Then, we can find in polynomial time a set X of $i \in [a-1]$ vertices and an independent set I with the following properties:*

(i) *The set X contains v.*

(ii) *The set $I \subseteq \bigcap_{x \in X} N(x)$ is an independent set of size at least $bk^{a-i} + 1$.*

(iii) *For every vertex $u \in V(G) \setminus X$, it holds that $|N(u) \cap I| \le bk^{a-i-1}$.*

Proof. By Lemma 6.7, there is an independent set I_v of size bk^{b-1} in $N(v)$ (which can be found in polynomial time). Let X be an inclusion-wise maximal set of i vertices including v with $|\bigcap_{x \in X} N(x) \cap I_v| > bk^{a-i}$. Such a set can be found by the following polynomial-time algorithm: We start with $X = \{v\}$ and $i = 1$. We will maintain the invariant that $|X| = i$. If there exists a vertex $v' \in V(G) \setminus X$ with $|N(v') \cap \bigcap_{x \in X} N(x) \cap I_v| > bk^{a-i-1}$, then we add v' to X and increase i by 1. We keep doing so until there remains no such vertex v'.

We show that this algorithm terminates for $i = |X| \le a - 1$. Assume to the contrary that the algorithm continues for $i = a-1$. We then have that $|N(v') \cap \bigcap_{x \in X} N(x) \cap I_v| > bk^{a-i-1}$ for some vertex $v' \in V(G) \setminus X$. It follows that the set $X \cup \{v'\}$ (which is of size a) has more than b common neighbors, contradicting the fact that G has no $K_{a,b}$ as a subgraph.

Finally, we show that a set X found by this algorithm and $I := \bigcap_{x\in X} N(x) \cap I_v$ satisfy the three properties of the lemma. We have $|I| = |\bigcap_{x\in X} N(x) \cap I_v| = |N(v') \cap \bigcap_{x\in X\setminus\{v'\}} N(x) \cap I_v| > bk^{a-(i-1)-1} = bk^{a-i}$, where v' is the last vertex added to X. Moreover, since X is inclusion-wise maximal, we have $|N(u) \cap I| = |N(u) \cap \bigcap_{x\in X} N(x) \cap I_v| \le bk^{a-i-1}$ for every vertex $u \in V(G) \setminus X$. ◀

▶ **Reduction Rule 6.9.** Let $\mathcal{I}$ be an instance of Annotated Degrading α-FCGP. Let X, I be as specified in Lemma 6.8 and let $v \in I$ be a vertex such that every other vertex in I is better than v. Then, apply the Exclusion Rule (Reduction Rule 3.3) to v.

▶ **Lemma 6.10.** *Reduction Rule 6.9 is correct.*

Proof. We show the proof for the maximization variant; the minimization variant follows analogously. For the sake of contradiction, assume that every solution S contains v. By Lemma 6.8, every vertex $u \in V(G) \setminus X$ has at most bk^{a-i} neighbors in I. Moreover, since I is an independent set, we have $|I \cap N[v']| = 1$ for every vertex $v' \in I$ (including v). For $S' := S \setminus X$, we have

$$\begin{aligned}|I \setminus N[S']| &\ge |I| - |I \cap N[v]| - |I \cap N[S' \setminus \{v\}]| \\ &\ge (bk^{a-i} + 1) - (k-1)bk^{a-i-1} - 1 = bk^{a-i-1} > 0.\end{aligned}$$

Let v' be an arbitrary vertex in $I \setminus N[S']$. We show that $\operatorname{cont}(v', S \setminus \{v\}) \ge \operatorname{cont}(v, S \setminus \{v\})$. By Lemma 3.5, this would imply that $(S \setminus \{v\}) \cup \{v'\}$ is a solution not containing v. Since v and v' are both adjacent to all vertices of X and $\alpha \in (1/3, 1]$, we have

$$\begin{aligned}\operatorname{cont}(v', S \setminus \{v\}) &= \alpha \deg^{+c}(v') + (1-3\alpha)|X \cap (S \setminus \{v\})| \\ &\ge \alpha \deg^{+c}(v) + (1-3\alpha)|X \cap (S \setminus \{v\})| \\ &\ge \alpha \deg^{+c}(v) + (1-3\alpha)|N(v) \cap (S \setminus \{v\})| = \operatorname{cont}(v, S \setminus \{v\}).\end{aligned}$$

Here, the first inequality follows from the fact that v' is better than v. ◀

By applying Reduction Rule 6.9 exhaustively, we end up with an instance with maximum degree $\Delta \le R(bk^{a-1})$. The following proposition then follows from Proposition 4.10 using the bound in Lemma 6.7:

▶ **Proposition 6.11.** *For any $a \le b \in \mathbb{N}$,* Degrading α-FCGP *on graphs that do not contain $K_{a,b}$ as a subgraph has a kernel of size $(R(bk^{a-1}) + k)^{\mathcal{O}(1)} \in b^{\mathcal{O}(a)} k^{\mathcal{O}(a^2)}$.*

Note that a d-degenerate graph contains no $K_{d+1,d+1}$ as a subgraph. We obtain the following result using the folklore fact that any d-degenerate graph on at least $(d+1)k$ vertices has an independent set of size k.

▶ **Lemma 6.12** ($\star$)**.** Degrading α-FCGP *admits a kernel of size $k^{\mathcal{O}(d)}$.*

Now, we show that significant improvement in Lemma 6.12 is unlikely. This, together with Lemma 6.12, implies Theorem 6.6.

▶ **Proposition 6.13** ($\star$)**.** Degrading Max α-FCGP *admits no kernel of size $\mathcal{O}(k^{d-2-\epsilon})$ unless* coNP $\subseteq$ NP/poly.

7 Parameterization by vc and h-index

To complete the picture of the parameterized complexity landscape, we consider two parameters that are larger than the degeneracy of G: the h-index of G and the vertex cover number of G. We start with the maximization variant and the h-index. As Non-Degrading Max α-FCGP does not admit a polynomial kernel with respect to k even if Δ is constant (see Thm. 4.11), the same holds for the h-index. The degrading case admits a polynomial kernel.

▶ **Proposition 7.1** ($\star$). Degrading Max α-FCGP *admits a kernel of size* $\mathcal{O}(h^2k^2 + k^4)$.

We complement this with showing fixed-parameter tractability for $k + h$.

▶ **Proposition 7.2** ($\star$). Non-Degrading Max α-FCGP *is fixed-parameter tractable with respect to* $k + h$.

For the larger parameter vertex cover number vc, we achieve a kernel for all $\alpha > 0$.

▶ **Proposition 7.3** ($\star$). *If* $\alpha \neq 0$, *then* Max α-FCGP *admits a kernel of size* $\mathcal{O}(\mathsf{vc}^4 + \mathsf{vc} \cdot k^3)$.

For $\alpha = 0$, Max α-FCGP corresponds to Densest k-Subgraph and Clique is one of its special cases ($t = \binom{k}{2}$). Since Clique does not admit a polynomial kernel with respect to vc [2] (and any clique is of size at most $\mathsf{vc} + 1$), Densest k-Subgraph does not admit a polynomial kernel. However, Densest k-Subgraph can be solved by a straightforward algorithm in $\mathcal{O}^*(2^{\mathsf{vc}})$ time. Thus, Densest k-Subgraph admits a kernel of size $\mathcal{O}(2^{\mathsf{vc}})$.

Minimization variant. Note that Degrading Min α-FCGP has a polynomial kernel with respect to $d + k$ (see Theorem 6.3) and, thus, also with respect to $h + k$ and $\mathsf{vc} + k$. As Non-Degrading Min α-FCGP does not admit a polynomial kernel with respect to k even if Δ is constant (see Thm. 4.11), the same holds for the h-index. It remains to consider Non-Degrading Min α-FCGP parameterized by $\mathsf{vc} + k$.

▶ **Proposition 7.4** ($\star$). Min α-FCGP *admits a kernel of size* $\mathcal{O}(\mathsf{vc}^8 + \mathsf{vc} \cdot k^7)$ *for* $\alpha > 0$ *and of size* $\mathcal{O}(\mathsf{vc}^2 + \mathsf{vc} \cdot k)$ *for* $\alpha = 0$.

8 Conclusion

We provided a systematic parameterized complexity analysis for α-FCGP (see Figure 2). Although we settled the existence of polynomial kernels with respect to various parameters combined with the solution size k, several open questions remain. First, our polynomial kernels are not optimized and thus the polynomials are of high degree. Looking for smaller kernels is thus an obvious first task. Second, can our positive results for c-closure and degeneracy be extended to the smaller parameter weak closure [16]? Furthermore, while we looked at parameters that are small in sparse graphs, can similar results be achieved for dense graphs as considered e. g. by Lochet et al. [27]? Finally, we believe that an experimental verification of our data reduction rules would demonstrate their practical usefulness.

References

1 Omid Amini, Fedor V. Fomin, and Saket Saurabh. Implicit Branching and Parameterized Partial Cover Problems. *Journal of Computer and System Sciences*, 77(6):1159–1171, 2011.

2 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240.

3 Edouard Bonnet, Bruno Escoffier, Vangelis Th. Paschos, and Emeric Tourniaire. Multi-parameter Analysis for Local Graph Partitioning Problems: Using Greediness for Parameterization. *Algorithmica*, 71(3):566–580, 2015.

4 Édouard Bonnet, Stéphan Thomassé, Xuan Thang Tran, and Rémi Watrigant. An Algorithmic Weakening of the Erdős-Hajnal Conjecture. In *Proceedings of the 28th Annual European Symposium on Algorithms (ESA '20)*, volume 173 of *LIPIcs*, pages 23:1–23:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

5 Nicolas Bourgeois, Aristotelis Giannakos, Giorgio Lucarelli, Ioannis Milis, and Vangelis Th. Paschos. Exact and superpolynomial approximation algorithms for the Densest k-Subgraph problem. *European Journal of Operational Research*, 262(3):894–903, 2017.

6 Maurizio Bruglieri, Matthias Ehrgott, Horst W. Hamacher, and Francesco Maffioli. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Applied Mathematics*, 154(9):1344–1357, 2006.

7 Leizhen Cai. Parameterized Complexity of Cardinality Constrained Optimization Problems. *The Computer Journal*, 51(1):102–121, 2008.

8 Leizhen Cai, Siu Man Chan, and Siu On Chan. Random Separation: A New Method for Solving Fixed-Cardinality Optimization Problems. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation (IWPEC '06)*, volume 4169 of *LNCS*, pages 239–250. Springer, 2006.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

10 Marek Cygan, Fabrizio Grandoni, and Danny Hermelin. Tight Kernel Bounds for Problems on Graphs with Small Degeneracy. *ACM Transactions on Algorithms*, 13(3):43:1–43:22, 2017.

11 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Kernelization hardness of connectivity problems in d-degenerate graphs. *Discrete Applied Mathematics*, 160(15):2131–2141, 2012.

12 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

13 Pavel Dvorák, Andreas Emil Feldmann, Ashutosh Rai, and Pawel Rzazewski. Parameterized Inapproximability of Independent Set in H-Free Graphs. In *Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science, (WG '20)*, volume 12301 of *Lecture Notes in Computer Science*, pages 40–53. Springer, 2020.

14 David Eppstein and Emma S. Spiro. The h-Index of a Graph and its Application to Dynamic Subgraph Statistics. *Journal of Graph Algorithms and Applications*, 16(2):543–567, 2012. doi:10.7155/jgaa.00273.

15 Uriel Feige and Michael Seltser. On the densest k-subgraph problem. Technical report, Weizmann Institute of Science. Department of Applied Mathematics and Computer Science, 1997.

16 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding Cliques in Social Networks: A New Distribution-Free Model. *SIAM Journal on Computing*, 49(2):448–464, 2020.

17 Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized Complexity of Vertex Cover Variants. *Theory of Computing Systems*, 41(3):501–520, 2007.

18 Satoshi Hara, Takayuki Katsuki, Hiroki Yanagisawa, Takafumi Ono, Ryo Okamoto, and Shigeki Takeuchi. Consistent and Efficient Nonparametric Different-Feature Selection. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS '17)*, volume 54 of *Proceedings of Machine Learning Research*, pages 130–138. PMLR, 2017.

19 Satoshi Hara, Tetsuro Morimura, Toshihiro Takahashi, Hiroki Yanagisawa, and Taiji Suzuki. A Consistent Method for Graph Based Anomaly Localization. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS '15)*, volume 38 of *Proceedings of Machine Learning Research*, pages 333–341. PMLR, 2015.

20 Stasys Jukna. *Extremal Combinatorics - With Applications in Computer Science*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2011.

21 Joachim Kneis, Alexander Langer, and Peter Rossmanith. Improved Upper Bounds for Partial Vertex Cover. In *Proceedings of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '08)*, volume 5344 of *LNCS*, pages 240–251, 2008.

22 Tomohiro Koana, Christian Komusiewicz, André Nichterlein, and Frank Sommer. Covering Many (or Few) Edges with k Vertices in Sparse Graphs. *CoRR*, abs/2201.05465, 2022. `arXiv:2201.05465`.

23 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Computing Dense and Sparse Subgraphs of Weakly Closed Graphs. In *Proceedings of the 31st International Symposium on Algorithms and Computation, (ISAAC '20)*, volume 181 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

24 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Exploiting c-Closure in Kernelization Algorithms for Graph Problems. In *Proceedings of the 28th Annual European Symposium on Algorithms (ESA '20)*, volume 173 of *LIPIcs*, pages 65:1–65:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

25 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Essentially Tight Kernels For (Weakly) Closed Graphs. In *Proceedings of the 32nd International Symposium on Algorithms and Computation, (ISAAC '21)*, volume 212 of *LIPIcs*, pages 35:1–35:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

26 Christian Komusiewicz and Manuel Sorge. An Algorithmic Framework for Fixed-Cardinality Optimization in Sparse Graphs Applied to Dense Subgraph Problems. *Discrete Applied Mathematics*, 193:145–161, 2015.

27 William Lochet, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Exploiting Dense Structures in Parameterized Complexity. In *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS '21)*, volume 187 of *LIPIcs*, pages 50:1–50:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.STACS.2021.50`.

28 Fahad Panolan and Hannane Yaghoubizade. Partial Vertex Cover on Graphs of Bounded Degeneracy. *CoRR*, abs/2201.03876, 2022. `arXiv:2201.03876`.

29 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial Kernels for Dominating Set in Graphs of Bounded Degeneracy and Beyond. *ACM Transactions on Algorithms*, 9(1):11:1–11:23, 2012.

30 Venkatesh Raman and Saket Saurabh. Short Cycles Make W-hard Problems Hard: FPT Algorithms for W-hard Problems in Graphs with no Short Cycles. *Algorithmica*, 52(2):203–225, 2008.

31 Saket Saurabh and Meirav Zehavi. $(k, n-k)$-MAX-CUT: An $\mathcal{O}^*(2^p)$-time algorithm and a polynomial kernel. *Algorithmica*, 80(12):3844–3860, 2018. `doi:10.1007/s00453-018-0418-5`.

32 Hadas Shachnai and Meirav Zehavi. Parameterized Algorithms for Graph Partitioning Problems. *Theory of Computing Systems*, 61(3):721–738, 2017. `doi:10.1007/s00224-016-9706-0`.

33 Rémi Watrigant, Marin Bougeret, and Rodolphe Giroudeau. Approximating the Sparsest k-Subgraph in Chordal Graphs. *Theory of Computing Systems*, 58(1):111–132, 2016.

One-To-Two-Player Lifting for Mildly Growing Memory

Alexander Kozachinskiy
Steklov Mathematical Institute of Russian Academy of Sciences, Moscow, Russia

Abstract

We investigate a phenomenon of “one-to-two-player lifting” in infinite-duration two-player games on graphs with zero-sum objectives. More specifically, let $\mathcal{C}$ be a class of strategies. It turns out that in many cases, to show that all two-player games on graphs with a given payoff function are determined in $\mathcal{C}$, it is sufficient to do so for one-player games. That is, in many cases the determinacy in $\mathcal{C}$ can be “lifted” from one-player games to two-player games. Namely, Gimbert and Zielonka (CONCUR 2005) have shown this for the class of positional strategies. Recently, Bouyer et al. (CONCUR 2020) have extended this to the classes of arena-independent finite-memory strategies. Informally, these are finite-memory strategies that use the same way of storing memory in all game graphs.

In this paper, we put the lifting technique into the context of memory complexity. The memory complexity of a payoff function measures, how many states of memory we need to play optimally in game graphs with up to n nodes, depending on n. We address the following question. Assume that we know the memory complexity of our payoff function in one-player games. Then what can be said about its memory complexity in two-player games? In particular, when is it finite?

In this paper, we answer this questions for strategies with “chromatic” memory. These are strategies that only accumulate sequences of colors of edges in their memory. We obtain the following results.

- Assume that the chromatic memory complexity in one-player games is sublinear in n on some infinite subsequence. Then the chromatic memory complexity in two-player games is finite.
- We provide an example in which ***(a)*** the chromatic memory complexity in one-player games is linear in n; ***(b)*** the memory complexity in two-player games is infinite.

Thus, we obtain the exact barrier for the one-to-two-player lifting theorems in the setting of chromatic finite-memory strategies. Previous results only cover payoff functions with constant chromatic memory complexity.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Games on graphs, one-to-two-player lifting, strategy complexity, positional determinacy, finite-memory determinacy

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.43

Related Version *Full Version*: arXiv:2104.13888 [17]

Funding This work was performed at the Steklov International Mathematical Center and supported by the Ministry of Science and Higher Education of the Russian Federation (agreement no. 075-15-2019-1614).

1 Introduction

We study two-player infinite-duration games on graphs. These games are of interest in many areas of computer science, ranging from purely theoretical disciplines, such as decidability of logical theories [22, 23], to more practically-oriented ones, such as controller synthesis [15].

These games are played as follows. There is a finite directed graph with a token. We will call this graph *arena*. Initially, the token is placed in one of the nodes of the arena. In each turn, one of the two players takes the token and moves it to some other node. A restriction is that there must be an edge to the new location of the token. For each node of the arena, it is fixed in advance which of the players is the one to move the token in this node. The game proceeds for infinitely many turns. The outcome of the game is decided by the resulting trajectory of the token (it forms an infinite path in the arena).

We restrict ourselves to *zero-sum* games. Correspondingly, the players will be called Max and Min from now on. In a zero-sum game, objectives of the players are defined through a *payoff function* – a function of the form $\varphi\colon C^\omega \to \mathcal{W}$, where C is a set of *colors*, and $(\mathcal{W}, \leq)$ is an arbitrary linearly ordered set. Next, we assume that arenas are edge-colored by elements of C. To compute the outcome of a play (which will be an element of $\mathcal{W}$), we take the trajectory of the token in this play, then consider the infinite sequence of colors $\gamma \in C^\omega$ written on the edges of the trajectory, and, finally, apply φ to γ. The aim of Max is to maximize $\varphi(\gamma)$, while the aim of Min is to minimize it (with respect to the ordering of $\mathcal{W}$).

As usually, a pair of strategies of the players in which the first strategy is the best response to the second one, and vice versa, is called an *equilibrium.* Next, a strategy which belongs to some equilibrium is called *optimal.* Now, a payoff function is called *determined* if in every arena there exists an equilibrium with respect to this payoff function.

We will study determinacy with respect to restricted classes of strategies. Namely, if $\mathcal{C}$ is a class of strategies, then we say that a payoff function is determined in $\mathcal{C}$ if the following holds: in every arena there is an equilibrium for this payoff function in which both strategies are from $\mathcal{C}$. The smaller is $\mathcal{C}$, the stronger is this requirement.

One of the main research directions in the area of games of graphs is *strategy complexity.* Its goal, broadly speaking, is to find out, for a payoff function φ of our interest, what is the "simplest" class of strategies $\mathcal{C}$ in which φ is determined. This is highly relevant when our task is to actually implement in practice one of the optimal strategies for φ. For instance, this is the case when we want to produce a device whose performance is measured by φ. If this device is meant to act in the environment, then the execution of this device can be modeled as a game – between the controller of the device and the environment. In this framework, the controller realizes one of the strategies in this game. Ideally, we want an optimal performance w.r.t. φ at the lowest cost (in terms of the resources we need to implement the controller). The lower is strategy complexity of φ, the easier is this task.

Classically, there are two classes of strategies that are often considered in this context. One is the class of *positional* strategies and the other is the class of *finite-memory* strategies.

Let us first consider positional strategies. A strategy is positional if, for every node v of the arena, it always makes the same move when the token is in v, no matter what was the path of the token to this node. Sometimes these strategies are called *memory-less* – they do not need to "remember" anything about the previous development of the game. For brevity, we call payoff functions that are determined in the class of positional strategies *positionally determined.* Classical examples of games with positionally determined payoff functions are Parity Games, Mean-Payoff Games and Discounted Games [21, 10, 24].

These games, especially Parity Games, had a tremendous impact on such areas as verification, model checking and program analysis [11, 12, 1]. However, say, in controller synthesis, it is often required to consider more complex games, namely, those for which positional strategies do not suffice. This brings us to a more general class of strategies – the class of finite-memory strategies.

Unlike positional strategies, finite-memory strategies can store some information about the previous development of the game. The point is that during the whole play, which is infinitely long, the amount of this information should never exceed some constant.

The storage of information in finite-memory strategies is carried out by *memory skeletons.* A memory skeleton $\mathcal{M}$ is a deterministic finite automaton whose input alphabet is the set of colors. Now, an $\mathcal{M}$-strategy is a strategy which, informally, stores information according to the memory skeleton $\mathcal{M}$. To understand how it works, imagine that during the game, each time the token is shifted along some edge, the color of this edge is fed to $\mathcal{M}$. Then, at every moment, the current state of $\mathcal{M}$ represents the current content of the memory. Correspondingly, the moves of an $\mathcal{M}$-strategy depend solely on the current state of $\mathcal{M}$ and the current node with the token.

A strategy is finite-memory if it is an $\mathcal{M}$-strategy for some memory skeleton $\mathcal{M}$. For brevity, we call payoff functions that are determined in the class of finite-memory strategies *finite-memory determined.*

▶ Remark 1. Finite-memory strategies as defined above are sometimes called “chromatic”. This is because one can consider a more general definition. Namely, one can allow memory skeletons to take the whole edge as an input, not only its color. However, as shown by Le Roux [18], determinacy in general finite-memory strategies is equivalent to determinacy in chromatic finite-memory strategies. In this paper, we work only with chromatic finite-memory strategies.

1.1 One-to-two-player lifting

One of the techniques in the area of strategy complexity is called *one-to-two-player lifting.* Our paper is devoted to this technique. It relies on the notion of *one-player arenas.* An arena is called one-player if for one of the players the following holds: all the nodes of the arena from which this player is the one to move have exactly one out-going edge. This means that one of the players is given no choice and has only one way of playing. Correspondingly, there are two types of one-player arenas – those in which Max has no choice and those in which Min has no choice.

It turns out that to study determinacy in some class of strategies $\mathcal{C}$, it is sometimes sufficient to consider only one-player arenas. As was shown by Gimbert and Zielonka [13], this applies to the class of positional strategies. More specifically, their result states the following. Assume that a payoff function is such that all *one-player* arenas have an equilibrium of two positional strategies[1] with respect to this payoff function. Then *all* arenas, not only one-player ones, have an equilibrium of two positional strategies with respect to this payoff function. That is, then this payoff function is positionally determined. In a way, this means the positional determinacy of one-player games can always be “lifted” to two-player games.

This result has fundamental significance for studying the positional determinacy. This is because often one-player arenas are considerably easier to analyze than two-player ones. Indeed, assume we have an arena in which, say, Min has no choice. A question of whether such an arena has a positional equilibrium reduces to the following question. Is there a “lasso” (a simple path to a simple cycle over which we rotate infinitely many times) which maximizes our payoff function over all infinite paths? Often this can be figured out with a simple graph reasoning. For instance, this is fairly easy for Parity Games and Mean Payoff Games. Thus, through the lifting theorem of Gimbert and Zielonka one gets simple proofs of positional determinacy of these games. In turn, proofs that existed prior to the paper of Gimbert and Zielonka were highly non-trivial.

Given such a success in the case of positional strategies, it is temping to extend this to larger classes of strategies. This was recently investigated for the class of finite-memory strategies by Bouyer et al. in [4]. It turns out that the situation is quite different for this

[1] Note that in one-player arenas, one of the players has just one strategy (and this strategy is positional). So this requirement means that the other player has a positional strategy which is at least as good against the unique strategy of the opponent as any other strategy.

class. More specifically, Bouyer et al. have constructed a payoff function such that ***(a)*** all one-player arenas have an equilibrium of two finite-memory strategies with respect to this payoff function ***(b)*** there is an arena (in fact, with just 2 nodes) which is not one-player and which has no equilibrium of two finite-memory strategies with respect to this payoff function.

Thus, the class of positional strategies admits one-to-two-player lifting and the class of finite-memory strategies does not. Bouyer et al. suggested to study intermediate classes. Namely, their approach was as follows. By definition, the class of finite-memory strategies is the union of the classes of $\mathcal{M}$-strategies over all memory skeletons $\mathcal{M}$. Let us now fix a memory skeleton $\mathcal{M}$ and consider the class of $\mathcal{M}$-strategies for this specific $\mathcal{M}$. Bouyer et al. show that for every $\mathcal{M}$ this class admits one-to-two-player lifting.

More precisely, the lifting theorem of Bouyer et al. states that for any memory skeleton $\mathcal{M}$ the following holds. Assume that a payoff function is such that all one-player arenas have an equilibrium of two $\mathcal{M}$-strategies. Then the same holds for all arenas, with exactly this memory skeleton $\mathcal{M}$. That is, then this payoff function is determined in $\mathcal{M}$-strategies.

Observe that positional strategies are exactly $\mathcal{M}$-strategies if the memory skeleton $\mathcal{M}$ has just one state. Thus, the lifting theorem Bouyer et al. includes the lifting theorem of Gimbert and Zielonka as a special case.

Bouyer et al. call payoff functions to which one can apply their lifting theorem *arena-independent finite-memory determined.* That is, a payoff function φ is arena-independent finite-memory determined if there exists a memory skeleton $\mathcal{M}$ such that φ is determined in $\mathcal{M}$-strategies.

In the literature there is a number of games with arena-independent finite-memory determined payoff functions. For example, one can list games with ω-regular winning conditions [7] and bounded multidimensional energy games [3]. In turn, unbounded multidimensional energy games are finite-memory determined but not arena-independently [9].

1.2 Our results

The aim of this work is to extend the lifting technique beyond the class of arena-independent finite-memory determined payoff functions.

For payoff functions beyond this class, there is no single memory $\mathcal{M}$ skeleton which suffices for all arenas (here "suffices" means the existence of an equilibrium of two $\mathcal{M}$-strategies). Instead, larger arenas require larger memory skeletons. This motivates a notion of the *memory complexity* of a payoff function. It can be defined as follows. For every n consider the minimal memory skeleton which is sufficient for all arenas with up to n nodes (w.r.t. our payoff function). Let the size of this memory skeleton (that is, the number of its states) be S_n. Then we call the function $n \mapsto S_n$ the memory complexity of our payoff function. Observe that arena-independent finite-memory determined payoff functions have memory complexity $O(1)$.

The memory complexity is the decisive factor in practice – if it grows too quickly, we might have no resources to implement optimal strategies for our payoff function. This complexity measure was studied for a number of payoff functions in [8, 9]

We initiate the study of the memory complexity in the context of one-to-two-player lifting. More specifically, we address the following question. Assume that we know the memory complexity of our payoff function in one-player arenas. Then what can be said about its memory complexity in all arenas? Thus, our approach differs from the approach of Bouyer et al. in the following regard. Instead of lifting determinacy in some fixed class of strategies from one-player arenas to all arenas, we lift bounds on the memory complexity.

To formulate our results, we introduce the following notation. Let $\mathbb{Z}^+$ denote the set of positive integers, and let $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ be a function. Then by $\mathsf{FMD}(f)$ we denote the class of all payoff functions φ such that for all $n \in \mathbb{Z}^+$ there exists a memory skeleton $\mathcal{M}$ with at most $f(n)$ states such that every arena with at most n nodes has an equilibrium of two $\mathcal{M}$-strategies with respect to φ. In other words, $\mathsf{FMD}(f)$ is the class of all payoff function with memory complexity at most f. We also introduce similar notation for one-player arenas. Namely, we let $\mathsf{1playerFMD}(f)$ be the class of all payoff functions φ such that for all $n \in \mathbb{Z}^+$ there exists a memory skeleton $\mathcal{M}$ with at most $f(n)$ states such that every *one-player* arena with at most n nodes has an equilibrium of two $\mathcal{M}$-strategies with respect to φ. Again, $\mathsf{1playerFMD}(f)$ is the class of payoff functions whose memory complexity in one-player arenas is at most f. Obviously, $\mathsf{FMD}(f) \subseteq \mathsf{1playerFMD}(f)$. Additionally, we let FMD stand for the class of all finite-memory determined payoffs. Finally, let $\mathsf{1playerFMD}$ be the class of all payoff functions φ such that every one-player arena has an equilibrium of two finite-memory strategies w.r.t. φ.

In this notation, the question we address in this paper can be formulated as follows: for which functions f and g do we have $\mathsf{1playerFMD}(f) \subseteq \mathsf{FMD}(g)$?

▶ Remark 2. One could consider an alternative definition of $\mathsf{FMD}(f)$, in which different arenas of size up to n may be mapped to different memory skeletons of size $f(n)$. Unfortunately, it is not clear how to extend results of this paper to this setting.

Before presenting our results, let us express previous ones in this notation. For technical convenience, we assume from now on that the set C of colors is finite. This is not an essential restriction, as any arena involves only finitely many colors. Hence, if C is infinite, one can study, separately all finite subsets $C' \subseteq C$, arenas that involve colors only from C'.

First, let us understand what payoff functions are included[2] in $\mathsf{FMD}(1)$. By definition, these are payoff functions such that for every n there is a memory skeleton $\mathcal{M}$ with 1 state such that all arenas with up to n nodes are determined in $\mathcal{M}$-strategies – or, equivalently, in positional strategies. Thus, $\mathsf{FMD}(1)$ is exactly the class of positionally determined payoff functions. Observe then that the lifting theorem of Gimbert and Zielonka can be stated as the equality $\mathsf{1playerFMD}(1) = \mathsf{FMD}(1)$.

In fact, the lifting theorem of Bouyer et al. asserts that, more generally, for any constant $k \in \mathbb{Z}^+$ we have $\mathsf{1playerFMD}(k) = \mathsf{FMD}(k)$. Indeed, take any $\varphi \in \mathsf{1playerFMD}(k)$. Our goal is to show that $\varphi \in \mathsf{FMD}(k)$. By definition, for every n there exists a memory skeleton $\mathcal{M}$ with at most k states such that all one-player arenas with at most n nodes have an equilibrium of two $\mathcal{M}$-strategies w.r.t. φ. A problem is that these $\mathcal{M}$ may be different for different n. However, since the set C of colors is finite, there are only finitely many memory skeletons with up to k states. One of them works for infinitely many n – and, hence, for all one-player arenas. Due to the lifting theorem of Bouyer et al., the same memory skeleton works for all arenas. Thus, since this memory skeleton has at most k states, we have $\varphi \in \mathsf{FMD}(k)$.

Let us note that the class of arena-independent finite-memory determined payoffs is the class $\mathsf{FMD}(O(1)) = \bigcup_{k\in\mathbb{Z}^+} \mathsf{FMD}(k)$.

[2] Here, formally, by $\mathsf{FMD}(1)$ we mean $\mathsf{FMD}(f)$ for the function $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ such that $f(n) = 1$ for all $n \in \mathbb{Z}^+$. More generally, if there is some expression in n defining a function $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$, we will use FMD of this expression instead of $\mathsf{FMD}(f)$. For example, if $f(n) = 2n^2 + 2$ for all $n \in \mathbb{Z}^+$, then we will write $\mathsf{FMD}(2n^2 + 2)$ instead of $\mathsf{FMD}(f)$.

Finally, since lifting does not hold for the whole class of finite-memory strategies, we have $\mathsf{1playerFMD} \neq \mathsf{FMD}$. In fact, this means that for some function f we have $\mathsf{1playerFMD}(f) \not\subseteq \mathsf{FMD}$. This is because

$$\mathsf{FMD} = \bigcup_f \mathsf{FMD}(f), \qquad \mathsf{1playerFMD} = \bigcup_f \mathsf{1playerFMD}(f)$$

over all $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$. Why is it so? For example, let us show this for FMD. We have to show that for any $\varphi \in \mathsf{FMD}$ and for every n there exists a memory skeleton $\mathcal{M}$ such that all arenas with up to n nodes have an equilibrium of two $\mathcal{M}$-strategies (w.r.t φ). A point is that, since C is finite, for every n the number of such arenas is also finite (w.l.o.g. we may assume that between each pair of nodes there are at most $|C|$ edges). In each of these arenas, fix a pair of finite-memory strategies forming an equilibrium (this is possible since $\varphi \in \mathsf{FMD}$). This gives a finite set of finite-memory strategies such that every arena with up to n nodes is determined in strategies from this set. It remains to set $\mathcal{M}$ to be the product of the memory skeletons of these strategies. Then all these strategies will be $\mathcal{M}$-strategies.

We proceed to our main result. Let $\Omega(n)$ denote the set of functions $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ for which there exists $C > 0$ such that $f(n) \geq Cn$ for all $n \in \mathbb{Z}^+$. We obtain the following lifting theorem:

▶ **Theorem 3.** *Consider any function* $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+, f \notin \Omega(n)$. *Define* $g\colon \mathbb{Z}^+ \to \mathbb{Z}^+, g(n) = f\left(\min\left\{m \mid \frac{f(m)}{m+1} \leq \frac{1}{2n}\right\}\right)$. *Then* $\mathsf{1playerFMD}(f) \subseteq \mathsf{FMD}(g)$.

First, why is the function g well-defined? Since $f \notin \Omega(n)$, the fraction $f(m)/m$ gets arbitrarily close to 0 for some m. Hence, the minimum in the definition of g is always over a non-empty set.

Now consider the case when, as in the lifting theorem of Bouyer et al., the function f is constant, that is $f(n) = k$ for some constant $k \in \mathbb{Z}^+$ and for all $n \in \mathbb{Z}^+$. Then we have $g(n) = k$ for all $n \in \mathbb{Z}^+$ as well. That is, our main results implies the equality $\mathsf{1playerFMD}(k) = \mathsf{FMD}(k)$, and this equality is the lifting theorem of Bouyer et al.

It is instructive to consider an example when $f \notin \Omega(n)$ and is super-constant. Say, assume that $f(n) = O(n^\gamma)$ for some $\gamma < 1$. It is easy to see that then $g(n) = O(n^{\gamma/(1-\gamma)})$. Now there is a gap between memory complexity in one-player arenas and in all arenas. The closer γ is to 1, the larger is this gap.

When γ gets equal to 1, Theorem 3 becomes inapplicable. We demonstrate that this is not due to the weakness of our technique.

▶ **Theorem 4.** $\mathsf{1playerFMD}(2n+2) \not\subseteq \mathsf{FMD}$.

This result shows the sharpness of Theorem 3. Namely, in order to obtain at least some bound on the memory complexity in all arenas, the memory complexity in one-player arenas should be a function not from $\Omega(n)$. In other words, it should be sublinear on some infinite subsequence. In turn, when it is already just linear, we might have no finite-memory determinacy.

Thus, our paper pushes the technique of one-to-two-player lifting to its limit. Unfortunately, this limit turns out to be very low. We are not aware of a payoff function which has been considered in the literature and to which one can apply Theorem 3, but which is not arena-independent finite-memory determined. For example, let us consider unbounded multidimensional energy games – as we have indicated, they are finite-memory determined but not arena-independently. As shown in [16], these games are in $\mathsf{FMD}(n^{O(1)})$. Here the constant in $O(1)$ depends on the dimension and the maximum of the norms of the weights. In any case, this bound is not sufficient for Theorem 3.

Still, we provide an example of a payoff function to which our lifting theorem is applicable and the lifting theorem of Bouyer et al. is not.

▶ **Theorem 5.** *There exists a function $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+, f \notin \Omega(n)$ and a payoff function from* $\mathsf{1playerFMD}(f)$ *which is not arena-independent finite-memory determined.*

1.3 Other related works and concluding remarks

First, the exact analogs of the theorems of Gimbert and Zielonka and Bouyer et al. for *stochastic* games were obtained in other works of these authors [14, 5]. We find it plausible that our result can be lifted to stochastic games as well. Le Roux and Pauly [19] obtained a *two-to-many-players* lifting theorem. Namely, they show that, under some conditions, two-player finite-memory determinacy implies that all multiplayer games have finite-memory Nash equilibrium. A different approach to study finite-memory determinacy can be found in [20].

A natural open question is to extend lifting theorems to strategies with non-chromatic finite memory. As we mentioned, Le Roux [18] has shown that non-chromatic finite memory can always be replaced by the chromatic one. Unfortunately, this transformation is rather costly – the size of the memory grows exponentially in the number of nodes. So even the following modest question seems to be open: is there a payoff function which has constant non-chromatic memory complexity in one-player games but is not finite-memory determined in two-player games?

Organization of the paper. In Section 2 we give preliminaries. In Section 3 we give brief overviews of the proofs of our results. The full proof of Theorem 3 is given in the Appendix B. The full proofs of Theorems 4 and 5 can be found in the arXiv version of this paper [17].

2 Preliminaries

Notation. We denote the set of positive integer numbers by $\mathbb{Z}^+$. Given a set A, by A^* and A^ω we denote the sets of finite and, respectively, infinite sequences of elements of A. The length of a sequence $x \in A^* \cup A^\omega$ is denoted by $|x|$. We write $A = B \sqcup C$ for three sets A, B, C if $A = B \cup C$ and $B \cap C = \varnothing$. Function composition is denoted by $\circ$.

2.1 Arenas

Following previous papers [13, 14, 4, 5], we call graphs on which our games are played *arenas*. We start with some notation regarding arenas. First, take an arbitrary finite set C. We will refer to the elements of C as *colors*. Informally, an arena is just a directed graph with edges colored by elements of C and with nodes partitioned into two sets.

▶ **Definition 6.** *A tuple $\mathcal{A} = \langle V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$, where*

- *$V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, E$ are four finite sets with $V = V_{\mathrm{Max}} \sqcup V_{\mathrm{Min}}$;*
- *$\mathsf{source}, \mathsf{target}, \mathsf{col}$ are functions of the form $\mathsf{source}\colon E \to V, \mathsf{target}\colon E \to V, \mathsf{col}\colon E \to C$;*

*is called an **arena** if for every $v \in V$ there exists $e \in E$ with $v = \mathsf{source}(e)$.*

Elements of V will be called **nodes** of $\mathcal{A}$ and elements of E will be called **edges** of $\mathcal{A}$. We understand $e \in E$ as a directed edge from the node $\mathsf{source}(e)$ to the node $\mathsf{target}(e)$. There might be parallel edges and loops. Additionally, every edge e of $\mathcal{A}$ is labeled by the color $\mathsf{col}(e) \in C$. Nodes from V_{Max} will be called nodes of Max and nodes from V_{Min} will be called nodes of Min. The out-degree of a node $v \in V$ is $|\{e \in E \mid \mathsf{source}(e) = v\}|$. By definition, every node in every arena has positive out-degree. An arena is called **one-player** if either all nodes of Max have out-degree 1 or all nodes of Min have out-degree 1.

Fix an arena $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$. We extend the function col (which determines the coloring of the edges) to arbitrary sequences of edges by setting: $\mathsf{col}(e_1 e_2 e_3 \ldots) = \mathsf{col}(e_1)\mathsf{col}(e_2)\mathsf{col}(e_3)\ldots$ for $e_1, e_2, e_3, \ldots \in E$.

A non-empty sequence of edges $h = e_1 e_2 e_3 \ldots \in E^* \cup E^\omega$ is called a **path** if for every $1 \le n < |h|$ we have $\mathsf{target}(e_n) = \mathsf{source}(e_{n+1})$. We define $\mathsf{source}(h) = \mathsf{source}(e_1)$. When h is finite, we define $\mathsf{target}(h) = \mathsf{target}(e_{|h|})$. In addition, for every $v \in V$ we consider a 0-length path λ_v identified with v, for which we set $\mathsf{source}(\lambda_v) = \mathsf{target}(\lambda_v) = v$. For every $v \in V$ we define $\mathsf{col}(\lambda_v)$ as the empty string.

2.2 Infinite-duration games on arenas

An arena $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ induces an infinite-duration two-player game in the following way. First, we call players of this game Max and Min. Informally, Max and Min interact by gradually constructing a longer and longer path in $\mathcal{A}$. In each turn one of the players extends a current path by some edge from its endpoint. Which of the two players is the one to move is determined by whether this endpoint belongs to V_{Max} or to V_{Min}.

Formally, positions in the game are finite paths in $\mathcal{A}$. By definition, $\mathsf{target}(h) \in V_{\text{Max}}$ for a finite path h means that Max is the one to move in the position h; respectively, $\mathsf{target}(h) \in V_{\text{Min}}$ means that Min is the one to move in the position h. A set of moves available in a position h is the set $\{e \in E \mid \mathsf{source}(e) = \mathsf{target}(h)\}$. Making a move $e \in E$ in a position $h = e_1 e_2 \ldots e_{|h|}$ brings to a position $he = e_1 e_2 \ldots e_{|h|} e$.

We stress that no position is designated as the initial one. We assume that the game can start in any position of the form $\lambda_v, v \in V$, at our choice.

Next we proceed to a notion of strategies. Namely, a strategy of Max is a function $\sigma\colon \{h \mid h \text{ is a finite path in } \mathcal{A} \text{ with } \mathsf{target}(h) \in V_{\text{Max}}\} \to E$ such that for every h from the domain of σ we have $\mathsf{source}(\sigma(h)) = \mathsf{target}(h)$. Respectively, a strategy of Min is a function $\tau\colon \{h \mid h \text{ is a finite path in } \mathcal{A} \text{ with } \mathsf{target}(h) \in V_{\text{Min}}\} \to E$ such that for every h from the domain of τ we have $\mathsf{source}(\tau(h)) = \mathsf{target}(h)$.

Observe that if $\mathcal{A}$ is one-player, then one of the players has exactly one strategy. For technical consistency we assume that even when one of the players owns all the nodes of $\mathcal{A}$, the other player still has one "empty" strategy.

A strategy induces a set of positions *consistent* with it (those that can be reached in a play against this strategy). Formally, a finite path $h = e_1 e_2 \ldots e_{|h|}$ is consistent with a strategy σ of Max if the following conditions hold:

- $\mathsf{source}(h) \in V_{\text{Max}} \implies \sigma(\lambda_{\mathsf{source}(h)}) = e_1$;
- for every $1 \le i < |h|$ we have $\mathsf{target}(e_1 e_2 \ldots e_i) \in V_{\text{Max}} \implies \sigma(e_1 e_2 \ldots e_i) = e_{i+1}$.

Consistency with the strategies of Min is defined similarly. Further, the notion of consistency can be extended to infinite paths. Namely, given a strategy, an infinite path is consistent with it if all finite prefixes of this path are.

For $v \in V$ and for a strategy $\mathcal{S}$ of one of the players $\mathsf{Cons}(v, \mathcal{S})$ denotes the set of all finite and infinite paths that start at v and are consistent with $\mathcal{S}$. For any strategy σ of Max, strategy τ of Min and $v \in V$, there is a unique infinite path in the intersection $\mathsf{Cons}(v, \sigma) \cap \mathsf{Cons}(v, \tau)$. We denote this path by $h(v, \sigma, \tau)$ and call it *the play* of σ and τ from v.

2.3 Payoff functions and equilibria

We consider only zero-sum games; correspondingly, in our framework objectives of the players are always given by a *payoff function.* A payoff function is any function of the form $\varphi\colon C^\omega \to \mathcal{W}$, where $(\mathcal{W}, \leq)$ is a linearly ordered set. Informally, the aim of Max is to play in a way which maximizes the payoff function (with respect to the ordering of $\mathcal{W}$) while the aim of Min is the opposite one. Technically, to get the value of the payoff function on a play (which is an infinite path in the underlying arena) we first apply the function col to this play; this gives us an infinite sequence of colors; in conclusion, we apply φ to the sequence of colors.

Any payoff function in a standard way induces a notion of an *equilibrium* of two strategies of the players (with respect to this payoff function). Let us first introduce a notion of an *optimal response.* Namely, take a strategy σ of Max and a strategy τ of Min. We say that σ is a **uniformly optimal response** to τ if for all $v \in V$ and for all infinite $h \in \mathsf{Cons}(v, \tau)$ we have $\varphi \circ \mathsf{col}\big(h(v, \sigma, \tau)\big) \geq \varphi \circ \mathsf{col}(h)$. The inequality here, of course, is with respect to the ordering of $\mathcal{W}$. Similarly, we call τ a **uniformly optimal response** to σ if for all $v \in V$ and for all infinite $h \in \mathsf{Cons}(v, \sigma)$ we have $\varphi \circ \mathsf{col}\big(h(v, \sigma, \tau)\big) \leq \varphi \circ \mathsf{col}(h)$. Next, we call a pair (σ, τ) a **uniform equilibrium** if σ and τ are uniformly optimal responses to each other.

▶ **Lemma 7.** *For any arena $\mathcal{A}$ and for any payoff function φ, the set uniform equilibria in $\mathcal{A}$ w.r.t. φ is a Cartesian product.*

Proof. See Appendix A. ◀

Strategies which belong to some uniform equilibrium will be called **uniformly optimal**.

▶ Remark 8. Each payoff function induces a total preoder on C^ω. Two payoff functions that induce the same preorder have the same set of equilibria. Due to this reason, previous papers in this line of work [13, 14, 4, 5] do not consider payoff functions at all. Instead, they directly consider total preorders on C^ω, to which they refer as *preference relations.* We prefer to use a terminology of payoff functions, as it is more standard. Of course, this does not make our results less general – any preference relation is induced by some payoff function.

2.4 Positional strategies and finite-memory strategies

Positional strategies. A strategy $\mathcal{S}$ of one of the players is called positional if for any two positions h_1, h_2 from its domain we have $\mathsf{target}(h_1) = \mathsf{target}(h_2) \implies \mathcal{S}(h_1) = \mathcal{S}(h_2)$. In other words, $\mathcal{S}(h)$ depends solely on $\mathsf{target}(h)$. It makes convenient to consider positional strategies as functions on the set of nodes of the corresponding players (rather than on the set of the positions of this player). I.e., positional strategies of Max can be identified with functions of the form $\sigma\colon V_{\mathrm{Max}} \to E$ such that $\mathsf{source}(\sigma(v)) = v$ for all $v \in V_{\mathrm{Max}}$. Similarly, positional strategies of Min can be identified with functions of the form $\tau\colon V_{\mathrm{Min}} \to E$ such that $\mathsf{source}(\tau(v)) = v$ for all $v \in V_{\mathrm{Min}}$.

Let us fix some notation regarding positional strategies. First, every edge $e \in E$ is a path (of length 1) and hence also a position in the game induced by $\mathcal{A}$. If $\mathcal{S}$ is a positional strategy of one of the players, we let $E_{\mathcal{S}}$ be the set of edges that are consistent with $\mathcal{S}$. Observe the following feature of positional strategies: the set of paths (positions) that are consistent with a positional strategy $\mathcal{S}$ is exactly the set of paths that consist only of edges from $E_{\mathcal{S}}$.

Given a positional strategy $\mathcal{S}$ of one of the players, by $\mathcal{A}_{\mathcal{S}}$ we denote the arena $\mathcal{A}_{\mathcal{S}} = \langle V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, E_{\mathcal{S}}, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$. That is, $\mathcal{A}_{\mathcal{S}}$ is obtained from $\mathcal{A}$ by deleting all edges that are inconsistent with $\mathcal{S}$. Observe that the arena $\mathcal{A}_{\mathcal{S}}$ is one-player; each node of the player who plays $\mathcal{S}$ has exactly one out-going edge in $\mathcal{A}_{\mathcal{S}}$.

Instead of saying "an equilibrium of two positional strategies" we will simply say "a positional equilibrium".

Finite-memory strategies. A memory skeleton is a deterministic finite automaton $\mathcal{M} = \langle M, m_{init} \in M, \delta\colon M \times C \to M\rangle$ whose input alphabet is the set C of colors. Here M is the set of states of $\mathcal{M}$, the state $m_{init} \in M$ is a designated initial state, and δ is the transition function of $\mathcal{M}$. By $|\mathcal{M}|$ we denote the number of states of a memory skeleton $\mathcal{M}$. Given $m \in M$, we extend $\delta(m, \cdot)$ to finite sequences of elements of C in a standard way. Now, a strategy $\mathcal{S}$ of one of the players is called an $\mathcal{M}$-strategy if for any two positions h_1 and h_2 from the domain of $\mathcal{S}$ it holds that

$$\big[\mathsf{target}(h_1) = \mathsf{target}(h_2) \text{ and } \delta(m_{init}, \mathsf{col}(h_1)) = \delta(m_{init}, \mathsf{col}(h_2))\big] \implies \mathcal{S}(h_1) = \mathcal{S}(h_2).$$

In other words, $\mathcal{S}(h)$ depends solely on $\mathsf{target}(h)$ (the node with the token in the position h) and $\delta(m_{init}, \mathsf{col}(h))$ (the state into which $\mathcal{M}$ comes after reading the sequence of colors along h).

A strategy $\mathcal{S}$ of one of the players is called a finite-memory strategy if it is an $\mathcal{M}$-strategy for some memory skeleton $\mathcal{M}$. Instead of saying "an equilibrium of two finite-memory strategies" or "an equilibrium of two $\mathcal{M}$-strategies" we will simply say "a finite-memory equilibrium" and "an $\mathcal{M}$-strategy equilibrium".

2.5 Determinacy and memory complexity

▶ **Definition 9.** *Let $\mathcal{C}$ be a class of strategies. We say that a payoff function φ is determined in $\mathcal{C}$ if every arena has a uniform equilibrium of two strategies from $\mathcal{C}$ w.r.t. φ. In particular,*

- *if $\mathcal{C}$ is the class of positional strategies, then we call φ* ***positionally determined****.*
- *if $\mathcal{C}$ is the class of finite-memory strategies, then we call φ* ***finite-memory determined****.*
- *if $\mathcal{C}$ is the class of $\mathcal{M}$-strategies for some memory skeleton $\mathcal{M}$, then we call φ* ***arena-independent finite-memory determined****.*

For our results it is important that we require equilibria to be uniform in these definitions. That is, it is important to have a single pair of strategies from $\mathcal{C}$ which is an equilibrium no matter in which node the game starts. As far as we know, this is the case for all positionally and finite-memory determined payoff functions that have been considered in the literature.

Next we provide definitions regarding the memory complexity.

▶ **Definition 10.** *Let* FMD *denote the class of functions $\varphi\colon C^\omega \to \mathcal{W}$ such that C is a finite set, $\mathcal{W}$ is linearly ordered and φ is finite-memory determined. Let* 1playerFMD *denote the class of functions $\varphi\colon C^\omega \to \mathcal{W}$ such that C is a finite set, $\mathcal{W}$ is a linearly ordered set and such that the following holds: every one-player arena (with edges colored by elements of C) has a uniform finite-memory equilibrium w.r.t. φ.*

Next, consider any function $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$. Let $\mathsf{FMD}(f)$ *denote the class of functions $\varphi\colon C^\omega \to \mathcal{W}$ such that C is a finite set, $\mathcal{W}$ is a linearly ordered set and such that the following holds: for all $n \in \mathbb{Z}^+$ there exists a memory skeleton $\mathcal{M}$ over the set C with $|\mathcal{M}| \le f(n)$ such that all arenas (with edges colored by elements of C) with at most n nodes have a uniform $\mathcal{M}$-strategy equilibrium w.r.t. φ. Similarly, let* $\mathsf{1playerFMD}(f)$ *denote the class of functions $\varphi\colon C^\omega \to \mathcal{W}$ such that C is a finite set, $\mathcal{W}$ is a linearly ordered set and such that the following holds: for all $n \in \mathbb{Z}^+$ there exists a memory skeleton $\mathcal{M}$ over the set C with $|\mathcal{M}| \le f(n)$ such that all one-player arenas (with edges colored by elements of C) with at most n nodes have a uniform $\mathcal{M}$-strategy equilibrium w.r.t. φ.*

3 Overviews of the Proofs

3.1 Theorem 3

First, let us give the exact statement of the lifting theorem of Bouyer et al.

▶ **Theorem 11** ([4]). *For any payoff function φ and for any memory skeleton $\mathcal{M}$ the following holds. Assume that all one-player arenas have a uniform $\mathcal{M}$-strategy equilibrium w.r.t. φ. Then all arenas have a uniform $\mathcal{M}$-strategy equilibrium w.r.t. φ.*

Our main technical contribution is the following strengthening of Theorem 11.

▶ **Theorem 12.** *For any payoff function φ and for any $n \in \mathbb{Z}^+$ the following holds. Let $\mathcal{M}$ be a memory skeleton such that all one-player arenas with at most $2n \cdot |\mathcal{M}| - 1$ nodes have a uniform $\mathcal{M}$-strategy equilibrium w.r.t. φ. Then all arenas with at most n nodes have a uniform $\mathcal{M}$-strategy equilibrium w.r.t. φ.*

Derivation of Theorem 3 from Theorem 12. Take any $\varphi \in \mathsf{1playerFMD}(f)$. Our goal is to show that $\varphi \in \mathsf{FMD}(g)$, where g is as in Theorem 3. That is, our goal is to establish for every $n \in \mathbb{Z}^+$ a memory skeleton $\mathcal{M}$ with at most $g(n)$ states such that all arenas with at most n nodes have a uniform $\mathcal{M}$-strategy equilibrium.

Take any $n \in \mathbb{Z}^+$. By definition, $g(n) = f(m)$ for some $m \in \mathbb{Z}$ such that $\frac{f(m)}{m+1} \leq \frac{1}{2n}$. Since $\varphi \in \mathsf{1playerFMD}(f)$, there exists a memory skeleton $\mathcal{M}$ with at most $f(m)$ states such that all one-player arenas with at most m nodes have a uniform $\mathcal{M}$-strategy equilibrium. Now, since $\frac{f(m)}{m+1} \leq \frac{1}{2n}$, we have $m \geq 2n \cdot f(m) - 1 \geq 2n \cdot |\mathcal{M}| - 1$. By Theorem 12, this means that all arenas with at most n nodes have a uniform $\mathcal{M}$-strategy equilibrium. Since $\mathcal{M}$ has at most $f(m) = g(n)$ states, we are done. ◀

Before discussing our technique, let us briefly overview how Bouyer et al. establish Theorem 11. They start by defining "$\mathcal{M}$-monotone payoff functions" and "$\mathcal{M}$-selective payoff functions". Then they show that any payoff function which is $\mathcal{M}$-monotone and $\mathcal{M}$-selective is determined in $\mathcal{M}$-strategies. Finally, they show that for any non-$\mathcal{M}$-monotone and for any non-$\mathcal{M}$-selective payoff function there exists a one-player arena which has no uniform $\mathcal{M}$-strategy equilibrium w.r.t. this payoff function. This also gives a *characterization* of $\mathcal{M}$-determinacy: a payoff function is determined in $\mathcal{M}$-strategies if and only if it is $\mathcal{M}$-monotone and $\mathcal{M}$-selective.

In this paper, we obtain Theorem 12 (and, thus, Theorem 11) more directly. For the sake of simplicity, in Section 4 we prove it in a special case when $\mathcal{M}$ is a memory skeleton with just one state. In this special case, $\mathcal{M}$-strategies are positional strategies.

▶ **Proposition 13** (Special case of Theorem 12). *For any payoff function φ and for any $N \in \mathbb{Z}^+$ the following holds. Assume that all one-player arenas with at most $2N - 1$ nodes have a uniform positional equilibrium w.r.t. φ. Then all arenas with at most N nodes have a uniform positional equilibrium w.r.t. φ.*

As all papers in this line of works, we build upon the inductive technique first invented by Gimbert and Zielonka [13]. Our contribution here is a more direct exposition of this technique, with the emphasis on the size of arenas.

We extend Proposition 13 to all memory skeletons[3] in two steps. We first prove an analogue of Proposition 13 for so-called $\mathcal{M}$-trivial arenas. Informally, these are arenas where states of $\mathcal{M}$ are "hardwired" into nodes. In such arenas, $\mathcal{M}$-strategies degenerate to positional strategies. We show that Proposition 13 is true even when only $\mathcal{M}$-trivial arenas are taken into account (in the assumption and in the conclusion).

We then derive Theorem 12 from this using the product arena construction [2, Chapter 2]. Take any (two-player) arena $\mathcal{A}$ with up to n nodes. We have to derive the existence of an $\mathcal{M}$-strategy equilibrium in $\mathcal{A}$ from the assumption of Theorem 12. It is a classical observation that $\mathcal{M}$-strategies in $\mathcal{A}$ can be viewed as positional strategies in the product arena $\mathcal{M} \times \mathcal{A}$. This product arena is obtained by first pairing states of $\mathcal{M}$ with nodes of $\mathcal{A}$, and then by drawing edges of $\mathcal{A}$ in all possible ways that are consistent with the transition function of $\mathcal{M}$. Now we only have to establish a positional equilibrium in $\mathcal{M} \times \mathcal{A}$. This arena is $\mathcal{M}$-trivial, so we use Proposition 13 for $\mathcal{M}$-trivial arenas and $N = n \cdot |\mathcal{M}|$. The size of $\mathcal{M} \times \mathcal{A}$ is the product of the sizes of $\mathcal{M}$ and $\mathcal{A}$, so it does not exceed N. It remains to show that all one-player $\mathcal{M}$-trivial arenas with up to $2N - 1 = 2n \cdot |\mathcal{M}| - 1$ nodes have a positional equilibrium. Indeed, by the assumption of Theorem 12, all one-player arenas (not only $\mathcal{M}$-trivial) of this size have an $\mathcal{M}$-strategy equilibrium. But in $\mathcal{M}$-trivial arenas these $\mathcal{M}$-strategy equilibria are automatically positional.

The full proof of Theorem 12 is given in Appendix B.

3.2 Theorem 4

Let the set of colors be $C = \{-1, 1\}$. We define a payoff function $\psi \colon C^\omega \to \{0, 1\}$ as follows. We set $\psi(c_1 c_2 c_3 \ldots) = 1$ if and only if either $\big(\lim_{n\to\infty} \sum_{i=1}^n c_i = +\infty\big)$ or $\big(\sum_{i=1}^n c_i = 0$ for infinitely many $n\big)$. We assume the standard ordering on $\{0, 1\} = \psi(C^\omega)$, so that 1 is interpreted as victory of Max and 0 is interpreted as victory of Min.

We show that $\psi \in \mathsf{1playerFMD}(2n+2) \setminus \mathsf{FMD}$. In fact, this payoff function was defined by Bouyer et al. in [4, Section 3.4]. They have shown that this payoff function is finite-memory determined in one-player arenas but not in two-player arenas. So our contribution here is an upper bound $\psi \in \mathsf{1playerFMD}(2n + 2)$ on its memory complexity in one-player arenas. In other words, for every n we provide a memory skeleton $\mathcal{M}_n$ with $2n + 2$ states such that every one-player arena $\mathcal{A}$ with up to n nodes has a uniform $\mathcal{M}_n$-strategy equilibrium. Let us describe the main ideas needed to obtain this upper bound. In this overview, we only consider those one-player arenas where all nodes of Min have out-degree 1. We use similar ideas for one-player arenas of the opposite type (but they require a bit more care).

It will be more convenient to refer to the elements of C as *weights* rather than as colors. Correspondingly, by the weight of a path we will mean the sum of the weights of its edges. Further, we will call a path *positive* if its weight is positive. We define negative and zero paths similarly.

Take an arena with up to n nodes where all nodes of Min have out-degree 1 (that is, essentially Max is the one to move everywhere). First, we can remove all the nodes from where one can reach a positive cycle. Indeed, Max has a positional winning strategy from these nodes (Max can go to the closest simple positive cycle, and then start rotating over it forever). Here it is important that our arena is one-player. Two-player arenas might have positive cycles, but Max might be unable to stay on them.

[3] Our technique in this part is rather similar to a technique from a recent paper of Bouyer et al. [5] (see the arXiv version [6] of their paper for more details). In this paper, they give a direct proof of an analogue of Theorem 11 for stochastic games.

Now the only way Max can win is by making the sum of the weights equal to 0 infinitely many times. As a first attempt, consider an "illegal" memory skeleton $\mathcal{M}_\infty$, which simply stores the sum of the weights along the current play. It is illegal since the sum of the weights can be arbitrarily large, so $\mathcal{M}_\infty$ will have infinitely many states. Still, our winning condition for Max can be reformulated in terms of $\mathcal{M}_\infty$. Indeed, Max just has to bring $\mathcal{M}_\infty$ into a state "the current sum is 0" infinitely many times. Notice that this is a parity condition in the product of our initial arena and the memory skeleton $\mathcal{M}_\infty$. Since parity games are positionally determined [23], we have a uniform positional equilibrium in the product arena, and this gives a uniform $\mathcal{M}_\infty$-strategy equilibrium in the initial arena.

To turn this idea into a proof, we "truncate" $\mathcal{M}_\infty$. For arenas with up to n nodes we consider a memory skeleton $\mathcal{M}_n$, which stores the current sum of the weights while its absolute value is at most n; if it exceeds n, our memory skeleton comes into a special invalid state. Observe that such memory skeleton requires just $2n + 2$ states.

We now make use of the fact that w.l.o.g our arena has no positive cycles. Since our weights are ±1, there is no path of weight larger than n. Indeed, any path can be decomposed into cycles and a simple path. The contribution of cycles is non-negative, and the contribution of a simple path is at most n, just because its length is at most n. So the current sum of the weights can never become larger than n. It can become smaller than $-n$, and in this case Max looses (he can never make it equal to 0 again). So the goal of Max is, first, to avoid a state "the current sum exceeded n in the absolute value", and second, to reach a state "the current sum is 0" infinitely many times. This is a parity condition in the product of our initial arena and the memory skeleton $\mathcal{M}_n$. Therefore, we get a uniform $\mathcal{M}_n$-strategy equilibrium in our initial arena.

3.3 Theorem 5

Let the set of colors be $C = \{0, 1\}$. Fix a set $T \subseteq \mathbb{Z}^+$. Define a payoff function $\varphi\colon \{0,1\}^\omega \to \{0, 1\}$ by setting $\varphi(\alpha) = 1$ for $\alpha = \alpha_1\alpha_2\alpha_3 \ldots \in \{0,1\}^\omega$ if and only if at least one of the following two conditions holds:

- α contains only finitely many 0's;
- for some $t \in T$, the sequence α contains the word 01^t0.

We show that, under some condition on T, the payoff function φ is not arena-independent finite-memory determined, but belongs to $\mathsf{1playerFMD}(f)$ for some $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+, f \notin \Omega(n)$. This condition is called *isolation*. Roughly speaking, it requires that there are infinitely many elements in T such that far to the left and to the right of them there are no other elements of T. More precisely, $T \subseteq \mathbb{Z}^+$ is isolated if there are infinitely many $k \in T$ such that $l \notin T$ for all $k/2 < l < k^4$, $l \neq k$. We call such k *isolated elements* of T.

From now on, we fix any isolated set T, for example, $T = \{2^{4^n} \mid n \in \mathbb{Z}^+\}$. To show that $\varphi \in \mathsf{1playerFMD}(f)$ for some $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+, f \notin \Omega(n)$, we construct, for every k, the following memory skeleton $\mathcal{M}_k$. It simply counts the number of 1's after the last 0. If this number exceeds k, it stops counting (it just remembers a fact that there are more than k ones after the last 0). Now, when our memory skeleton receives a 0, there are two cases. If the current value of the counter is some number from $T \cap [1, k]$, then $\mathcal{M}_k$ transits into a special "winning state", and stays in it forever. Otherwise, it resets the counter to 0 and starts counting again.

Note that $\mathcal{M}_k$ can be realized with $k+O(1)$ states. We show that if k is an isolated element of T, then all arenas (even two-player) with up to k^2 nodes have a uniform $\mathcal{M}_k$-strategy equilibrium. This will show that $\varphi \in \mathsf{FMD}(f)$ for some function f such that $f(n) \leq 2\sqrt{n}$ for infinitely many n.

Consider any arena with up to k^2 nodes. We define an auxiliary game in which Max wins if either $\mathcal{M}_k$ was brought to the "winning state" or there were just finitely many 0's in the play. Note that if Max wins in the auxiliary game, then Max wins w.r.t. φ. The auxiliary game, however, is not entirely equivalent to φ, because a play can be winning for Min in the auxiliary game but loosing for Min w.r.t. φ (if this play contains 01^t0 for some $t \in T, t > k$). Still, it holds that if Min can win in the auxiliary game, then Min can also win w.r.t. φ. To prove this claim, we notice that the auxiliary game is a parity game in the product of our initial arena and the memory skeleton $\mathcal{M}_k$. So if Min can win in it, then Min can do so via some positional strategy τ in the product arena. We observe that τ is also winning w.r.t. φ. Indeed, otherwise there is a play against τ which contains 01^t0 for some $t \in T, t > k$. Since k is an isolated element of T, we have $t \geq k^4$. Therefore, as the size of the product arena is $(k + O(1)) \cdot k^2 < k^4$, there must be a cycle which is consistent with τ and which consists entirely of 1's. But then Max can win against τ in the auxiliary game, contradiction.

As we pointed out, the auxiliary game is a parity game in the product of our arena with $\mathcal{M}_k$. Thus, it has a positional equilibrium there. This positional equilibrium translates into an $\mathcal{M}_k$-strategy equilibrium in the initial arena. Finally, as shown in the previous paragraph, any equilibrium in the auxiliary game is also an equilibrium w.r.t. φ.

Showing that φ is not arena-independent finite-memory determined is much easier. Take an isolated element $k \in T$. The idea is to construct an arena with a node which "cuts" the word 01^k0 in $\Omega(k)$ different ways near the middle. Due to isolation, the only way for Max to win in this arena is to go through one of the cuts. However, Min can choose any of the cuts, so Max needs $\Omega(k)$ states to distinguish between different cuts. Since k can be arbitrarily large, this shows that no single memory skeleton can be sufficient for φ in all arenas.

4 Proof of Proposition 13

The proof is by induction on the number of edges of an arena. More precisely, we are proving by induction on m the following claim: for every m every arena with m edges and at most N nodes has a uniform positional equilibrium.

The induction base ($m = 1$) is trivial (any arena with one edge is one-player and has exactly one node, so we can just refer to the assumption of the lemma). We proceed to the induction step. Take an arena $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ with at most N nodes and assume that all arenas with at most N nodes and with fewer edges than $\mathcal{A}$ have a uniform positional equilibrium. We prove the same for $\mathcal{A}$. Since the set of uniform equilibria is a Cartesian product by Lemma 7, it is enough to establish the following two claims:

- ***(a)*** in $\mathcal{A}$ there exists a uniform equilibrium including a positional strategy of Max;
- ***(b)*** in $\mathcal{A}$ there exists a uniform equilibrium including a positional strategy of Min.

We only show ***(a)***, a proof of ***(b)*** is similar.

We may assume that $\mathcal{A}$ is not one-player (otherwise we are done due to the assumptions of the lemma). Hence there exists a node $w \in V_{\text{Max}}$ with out-degree at least 2. Partition the set $E(w) = \{e \in E \mid \mathsf{source}(e) = w\}$ into two non-empty disjoint subsets $E_1(w)$ and $E_2(w)$. Define two new arenas $\mathcal{A}_1$ and $\mathcal{A}_2$. The arena $\mathcal{A}_1$ is obtained from $\mathcal{A}$ by deleting edges from the set $E_2(w)$. Similarly, the arena $\mathcal{A}_2$ is obtained from $\mathcal{A}$ by deleting edges from the set $E_1(w)$. So in $\mathcal{A}_i$ for $i = 1, 2$ the set of edges with the source in w is $E_i(w)$.

Both $\mathcal{A}_1$ and $\mathcal{A}_2$ have fewer edges than $\mathcal{A}$. So both these arenas have a uniform positional equilibrium. Let (σ_i, τ_i) be a uniform positional equilibrium in $\mathcal{A}_i$ for $i = 1, 2$. We will first define two auxiliary strategies τ_{12} and τ_{21} of Min; then we will show that either (σ_1, τ_{12}) or (σ_2, τ_{21}) is a uniform equilibrium in $\mathcal{A}$. After that ***(a)*** will be proved.

Strategies τ_{12} and τ_{21} will not be positional. In a sense, they are combinations of τ_1 and τ_2. In both strategies Min has a counter I which can only take two values, 1 and 2. The counter I indicates to Min which of the strategies τ_1 or τ_2 to use. I.e., whenever Min should make a move from a node $v \in V_{\mathrm{Min}}$, he uses an edge $\tau_I(v)$. The value of I changes each time in the node w Max uses an edge not from a set $E_I(w)$. It only remains to specify the initial value of I. There are two ways to do this, one will give us strategy τ_{12}, and the other will give τ_{21}. More specifically, in τ_{12} the initial value of I is 1 and in τ_{21} the initial value of I is 2.

It is not hard to see that τ_{12} is a uniformly optimal response to σ_1 and τ_{21} is a uniformly optimal response to σ_2. For instance, let us show this for τ_{12} and σ_1. By definition, τ_1 is a uniformly optimal response to σ_1 in the arena $\mathcal{A}_1$, and hence also in the arena $\mathcal{A}$ (because any play against σ takes place inside $\mathcal{A}_1$). It remains to notice that τ_{12} plays exactly as τ_1 against σ_1. Indeed, σ_1 never uses edges from $E_2(w)$, so the counter I always equals 1 against σ_1.

It remains to show that either σ_1 is a uniformly optimal response to τ_{12} or σ_2 is a uniformly optimal response to τ_{21} (in the arena $\mathcal{A}$). We derive it from the assumption of the lemma applied to an auxiliary one-player arena $\mathcal{B}$ with at most $2N-1$ nodes.

Namely, we define $\mathcal{B}$ as follows. Recall that in our notation $(\mathcal{A}_1)_{\tau_1}$ and $(\mathcal{A}_2)_{\tau_2}$ stand for two arenas obtained from, respectively, $\mathcal{A}_1$ and $\mathcal{A}_2$ by throwing away edges that are inconsistent with, respectively, τ_1 and τ_2. Consider an arena consisting of two "independent" parts one of which coincides with $(\mathcal{A}_1)_{\tau_1}$ and the other with $(\mathcal{A}_2)_{\tau_2}$ ("independent" means that there are no edges between the parts). From each part take a node corresponding to the node w. Then merge these two nodes into a single one. The resulting arena with $2|V|-1 \leq 2N-1$ nodes will be $\mathcal{B}$.

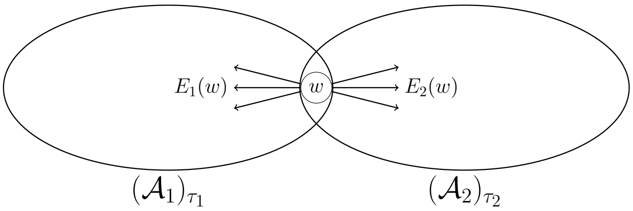


Figure 1 Arena $\mathcal{B}$.

For each node of $\mathcal{A}$ there are two "copies" of it in $\mathcal{B}$ – one from $(\mathcal{A}_1)_{\tau_1}$ and the other from $(\mathcal{A}_2)_{\tau_2}$. We will call copies of the first kind *left copies* and copies of the second kind *right copies*. Note that the left and the right copy of w is the same node in $\mathcal{B}$. Any other node of $\mathcal{A}$ has two distinct copies. Now, by the *prototype* of a node v' of $\mathcal{B}$ we mean a node v of $\mathcal{A}$ of which v' is a copy.

Note that in $\mathcal{B}$ all nodes of Min have out-degree 1 (because they do so inside $(\mathcal{A}_1)_{\tau_1}$ and $(\mathcal{A}_2)_{\tau_2}$, and the only node of $\mathcal{B}$ which was obtained by merging two nodes is a node of Max). Thus, $\mathcal{B}$ is a one-player arena.

An important feature of $\mathcal{B}$ is that it can "emulate" any play against τ_{12} and τ_{21} in $\mathcal{A}$. Formally,

▶ **Lemma 14.** *For any infinite path h in $\mathcal{A}$ which is consistent with τ_{12} there exists an infinite path h' in $\mathcal{B}$ with $\mathsf{col}(h') = \mathsf{col}(h)$ and with the source in the left copy of $\mathsf{source}(h)$. Similarly, for any infinite path h in $\mathcal{A}$ which is consistent with τ_{21} there exists an infinite path h' in $\mathcal{B}$ with $\mathsf{col}(h') = \mathsf{col}(h)$ and with the source in the right copy of $\mathsf{source}(h)$.*

Proof. We only give an argument for τ_{12}, the argument for τ_{21} is similar. We construct h' from the left copy of $\mathsf{source}(h)$ by always moving in the same "local direction" as h. There will be no problem with that for the nodes of Max because they have the same set of out-going edges in $\mathcal{B}$ as their prototypes have in $\mathcal{A}$. Now, for the nodes of Min we should be more accurate. The path h is consistent with τ_{12}, so from the nodes of Min it applies either τ_1 or τ_2. Now, in $\mathcal{B}$ strategy τ_1 is available only in the left ellipse of Figure 1, and τ_2 is available only in the right ellipse. So each time h wants to apply τ_1, the path h' should be in the left ellipse. Similarly, each time h wants to apply τ_2, the path h' should be in the right ellipse. Initially, until its counter changes, τ_{12} applies τ_1, and correspondingly h' starts in the left ellipse. Now, each time τ_{12} switches to τ_2, it does so because Max used an edge from $E_2(w)$ in w. Correspondingly, h' enters the right ellipse at this moment. Similarly, whenever τ_{12} switches back to τ_1, the path h' returns to the left ellipse. ◀

Note that in $\mathcal{B}$ Min has exactly one strategy. We denote it by T. The arena $\mathcal{B}$ is one-player and has at most $2N-1$ nodes, so by the assumption of the lemma there is a uniform positional equilibrium $(\widehat{\Sigma}, T)$ in it. We claim the following:

- if $\widehat{\Sigma}$ applies an edge from $E_1(w)$ in w, then σ_1 is a uniformly optimal response to τ_{12} in $\mathcal{A}$;
- if $\widehat{\Sigma}$ applies an edge from $E_2(w)$ in w, then σ_2 is a uniformly optimal response to τ_{21} in $\mathcal{A}$.

We only show the first claim, the proof of the second one is analogous. Consider a restriction of $\widehat{\Sigma}$ to the left ellipse of $\mathcal{B}$. This defines a positional strategy $\widehat{\sigma}$ of Max in $\mathcal{A}$. Note that in each node of $\mathcal{A}$ the strategy σ_1 is at least as good against τ_{12} as $\widehat{\sigma}$. Indeed, $\sigma_1(w), \widehat{\sigma}(w) \in E_1(w)$. Hence $\sigma_1, \widehat{\sigma}$ are strategies in the arena $\mathcal{A}_1$, where σ_1 is a uniformly optimal response to τ_1. It remains to notice that τ_{12} plays exactly as τ_1 against σ_1 and $\widehat{\sigma}$ since these two strategies of Max never use edges from $E_2(w)$.

Therefore, it is enough to show that $\widehat{\sigma}$ is a uniformly optimal response to τ_{12} in $\mathcal{A}$. Take any node $v \in V$ and any play h against τ_{12} from v. Our goal is to show that the play of $\widehat{\sigma}$ and τ_{12} from v is at least as good from the Max's perspective as h. Now, by Lemma 14 some infinite path h' from the left copy of v is colored exactly as h. On the other hand, the play of $\widehat{\Sigma}$ and T from the left copy of v is at least as good for Max as h' (and hence as h). This is because h' is consistent with T (as there are simply no other strategies of Min in $\mathcal{B}$) and because $(\widehat{\Sigma}, T)$ is an equilibrium. It remains to note that the play of $\widehat{\Sigma}$ and T from the left copy of v is colored exactly as the play of $\widehat{\sigma}$ and τ_{12} from v. Indeed, as we have already observed, τ_{12} plays exactly as τ_1 against $\widehat{\sigma}$. On the other hand, the play of $\widehat{\Sigma}$ and T can never leave the left ellipse as $\widehat{\Sigma}$ points to the left in w. Moreover, restrictions of these strategies to the left ellipse coincide with $\widehat{\sigma}$ and τ_1; for $\widehat{\Sigma}$ this is just by definition and for T this is because the left ellipse coincides with the arena $(\mathcal{A}_1)_{\tau_1}$.

References

1 Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games on continuous lattices. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

2 Mikołaj Bojańczyk and Wojciech Czerwiński. An automata toolbox. A book of lecture notes, available at `https://www.mimuw.edu.pl/~bojan/upload/reduced-may-25.pdf`, 2018.

3 Patricia Bouyer, Uli Fahrenberg, Kim G Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 33–47. Springer, 2008.

4 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

5 Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-Independent Finite-Memory Determinacy in Stochastic Games. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory (CONCUR 2021)*, volume 203 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CONCUR.2021.26.

6 Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. *arXiv preprint*, 2021. arXiv:2102.10104.

7 J Richard Büchi and Lawrence H Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

8 Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60, 2012.

9 Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta informatica*, 51(3-4):129–163, 2014.

10 Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

11 E Allen Emerson and Charanjit S Jutla. Tree automata, mu-calculus and determinacy. In *FoCS*, volume 91, pages 368–377. Citeseer, 1991.

12 E Allen Emerson, Charanjit S Jutla, and A Prasad Sistla. On model-checking for fragments of μ-calculus. In *International Conference on Computer Aided Verification*, pages 385–396. Springer, 1993.

13 Hugo Gimbert and Wiesław Zielonka. Games where you can play optimally without any memory. In *International Conference on Concurrency Theory*, pages 428–442. Springer, 2005.

14 Hugo Gimbert and Wieslaw Zielonka. Pure and stationary optimal strategies in perfect-information stochastic games with global preferences. *arXiv preprint*, 2016. arXiv:1611.08487.

15 Erich Gradel and Wolfgang Thomas. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer Science & Business Media, 2002.

16 Marcin Jurdziński, Ranko Lazić, and Sylvain Schmitz. Fixed-dimensional energy games are in pseudo-polynomial time. In *International Colloquium on Automata, Languages, and Programming*, pages 260–272. Springer, 2015.

17 Alexander Kozachinskiy. One-to-two-player lifting for mildly growing memory. *arXiv preprint*, 2021. arXiv:2104.13888.

18 Stéphane Le Roux. Time-aware uniformization of winning strategies. In *Conference on Computability in Europe*, pages 193–204. Springer, 2020.

19 Stéphane Le Roux and Arno Pauly. Extending finite-memory determinacy to multi-player games. *Information and Computation*, 261:676–694, 2018.

20 Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by boolean combination of winning conditions. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2018.

21 A Mostowski. Games with forbidden positions. Technical report, Preprint No. 78, Uniwersytet Gdanski, Instytyt Matematyki, 1991.

22 An A Muchnik. Games on infinite trees and automata with dead-ends: a new proof for the decidability of the monadic second order theory of two successors. *Bulletin-European Association For Theoretical Computer Science*, 48:219–219, 1992.

23 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

24 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.

A Uniform equilibria and T-wise equilibria

For the proof of Theorem 12 we need to generalize the notion of a uniform equilibrium. Take any arena $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ and any payoff function $\varphi\colon C^\omega \to \mathcal{W}$. Fix a subset $T \subseteq V$, a strategy σ of Max and a strategy τ of Min. We say that σ is a **T-wise optimal response** to τ w.r.t. φ if for all $v \in T$ and for all infinite $h \in \mathsf{Cons}(v, \tau)$ we have $\varphi\circ\mathsf{col}\big(h(v,\sigma,\tau)\big) \geq \varphi\circ\mathsf{col}(h)$. Similarly, we call τ a **T-wise optimal response** to σ w.r.t. φ if for all $v \in T$ and for all infinite $h \in \mathsf{Cons}(v, \sigma)$ we have $\varphi\circ\mathsf{col}\big(h(v,\sigma,\tau)\big) \leq \varphi\circ\mathsf{col}(h)$. Finally, we call a pair (σ, τ) a **T-wise equilibrium** w.r.t. φ if σ and τ are T-wise optimal responses to each other.

When $T = V$ is the whole set of nodes, then T-wise equilibria are uniform equilibria, and vice versa. Thus, the following lemma generalizes Lemma 7.

▶ **Lemma 15.** *For any arena $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$, for any payoff function φ, and for any subset $T \subseteq V$, the set of T-wise equilibria in $\mathcal{A}$ w.r.t. φ is a Cartesian product.*

Proof. It is sufficient to show the following: if (σ_1, τ_1) and (σ_2, τ_2) are T-wise equilibria, then so is (σ_1, τ_2). That is, our goal is to show that σ_1 is a T-wise optimal response to τ_2, and that τ_2 is a T-wise optimal response to σ_1. We only prove the first claim, the second one can be proved similarly. Take any $v \in T$ and any infinite $h \in \mathsf{Cons}(v, \tau_2)$. We have to show that $\varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_2)\big) \geq \varphi\circ\mathsf{col}(h)$. We first show that $\varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_2)\big) = \varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_1)\big) = \varphi\circ\mathsf{col}\big(h(v,\sigma_2,\tau_2)\big)$. Indeed,

$$\begin{aligned}\varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_1)\big) &\geq \varphi\circ\mathsf{col}\big(h(v,\sigma_2,\tau_1)\big) \geq \varphi\circ\mathsf{col}\big(h(v,\sigma_2,\tau_2)\big)\\ &\geq \varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_2)\big) \geq \varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_1)\big).\end{aligned}$$

The first inequality here holds because σ_1 is a T-wise optimal response to τ_1. The second inequality here holds because τ_2 is a T-wise optimal response to σ_2. The third inequality here holds because σ_2 is a T-wise optimal response to τ_2. The fourth inequality here holds because τ_1 is a T-wise optimal response to σ_1.

As we have shown, $\varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_2)\big) = \varphi\circ\mathsf{col}\big(h(v,\sigma_2,\tau_2)\big)$. In turn, since $h \in \mathsf{Cons}(v, \tau_2)$, and since σ_2 is a T-wise optimal response to τ_2, we have that $\varphi\circ\mathsf{col}\big(h(v,\sigma_2,\tau_2)\big) \geq \varphi\circ\mathsf{col}(h)$. Therefore, we get $\varphi\circ\mathsf{col}\big(h(v,\sigma_1,\tau_2)\big) \geq \varphi\circ\mathsf{col}(h)$. ◀

B Proof of Theorem 12

We reduce Theorem 12 to a statement about positional strategies (namely, to Lemma 20 below). First we need a classical concept of *product arenas.*

▶ **Definition 16** (Product arenas). *Let $\mathcal{M} = \langle M, m_{init}, \delta\colon M \times C \to M\rangle$ be a memory skeleton and $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ be an arena. Then $\mathcal{M} \times \mathcal{A}$ stands for an arena, where*

- *the set of nodes is $M \times V$;*
- *the set of Max's nodes is $M \times V_{\text{Max}}$;*
- *the set of Min's nodes is $M \times V_{\text{Min}}$;*
- *the set of edges is $M \times E$;*
- *the source function is defined as follows: $\mathsf{source}((m, e)) = (m, \mathsf{source}(e))$;*
- *the target function is defined as follows: $\mathsf{target}((m, e)) = \big(\delta(m, \mathsf{col}(e)), \mathsf{target}(e)\big)$;*
- *the coloring function is defined as follows: $\mathsf{col}((m, e)) = \mathsf{col}(e)$.*

The following is a standard observation that product arenas reduce finite-memory determinacy to positional determinacy.

▶ **Observation 17.** *Let $\mathcal{M} = \langle M, m_{init}, \delta\colon M \times C \to M\rangle$ be a memory skeleton and $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ be an arena. Then for every $S \subseteq V$ the following holds: if $\mathcal{M} \times \mathcal{A}$ has an $(\{m_{init}\} \times S)$-wise positional equilibrium, then $\mathcal{A}$ has an S-wise $\mathcal{M}$-strategy equilibrium.*

Its full proof can be found in the arXiv version of this paper [17].

Next we introduce one more concept which we need for the reduction, namely, one of *$\mathcal{M}$-triviality.*

▶ **Definition 18.** *Let $\mathcal{M} = \langle M, m_{init}, \delta\colon M \times C \to M\rangle$ be a memory skeleton. A pair $(\mathcal{A}, f)$ of an arena $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ and a function $f\colon V \to M$ is called **$\mathcal{M}$-trivial** if for every $e \in E$ it holds that $\delta\big(f(\mathsf{source}(e)), \mathsf{col}(e)\big) = f(\mathsf{target}(e))$.*

Informally, f is a mapping from $\mathcal{A}$ to the transition graph of $\mathcal{M}$ which takes into account the colors of the edges. Of course, there are arenas that belong to no $\mathcal{M}$-trivial pair. We observe that $\mathcal{M}$-strategies, in a sense, degenerate to positional ones in $\mathcal{M}$-trivial pairs.

▶ **Observation 19.** *Let $\mathcal{M} = \langle M, m_{init}, \delta\colon M \times C \to M\rangle$ be a memory skeleton. Then for every $\mathcal{M}$-trivial pair $(\mathcal{A}, f)$ the following holds: if $\mathcal{A}$ has a uniform $\mathcal{M}$-strategy equilibrium, then $\mathcal{A}$ has an $f^{-1}(m_{init})$-wise positional equilibrium.*

Proof. Note that for any finite path h in $\mathcal{A}$ we have:

$$\delta(f(\mathsf{source}(h)), \mathsf{col}(h)) = f(\mathsf{target}(h)).$$

Indeed, this holds by definition as long as h is a single edge; for longer h this can be easily proved by induction on $|h|$.

To show the observation, we simply show that any $\mathcal{M}$-strategy coincides with some positional one on all plays that start in the nodes of $f^{-1}(m_{init})$. Indeed, a move of an $\mathcal{M}$-strategy in a position h depends solely on $\mathsf{target}(h)$ and $\delta(m_{init}, \mathsf{col}(h))$. However, $\delta(m_{init}, \mathsf{col}(h)) = \delta\big(f(\mathsf{source}(h)), \mathsf{col}(h)\big) = f(\mathsf{target}(h))$ for all h with $\mathsf{source}(h) \in f^{-1}(m_{init})$. In other words, for all such h a move of an $\mathcal{M}$-strategy in h is a function only of $\mathsf{target}(h)$, as required. ◀

We are ready to formulate a statement about positional strategies to which we reduce Theorem 12.

▶ **Lemma 20.** *Let $\mathcal{M} = \langle M, m_{init}, \delta\colon M \times C \to M\rangle$ be a memory skeleton. Assume that for every $\mathcal{M}$-trivial pair $(\mathcal{A}, f)$ such that $\mathcal{A}$ is one-player and has at most $2N - 1$ nodes there exists an $f^{-1}(m_{init})$-wise positional equilibrium in $\mathcal{A}$.*

Then for every $\mathcal{M}$-trivial pair $(\mathcal{A}, f)$ such that $\mathcal{A}$ has at most N nodes there exists an $f^{-1}(m_{init})$-wise positional equilibrium in $\mathcal{A}$.

Derivation of Theorem 12 from Lemma 20. Let $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ be an arena with at most n nodes. Our goal is to show that $\mathcal{A}$ has a uniform $\mathcal{M}$-strategy equilibrium. By Observation 17, it is sufficient to show that the arena $\mathcal{M} \times \mathcal{A}$ has an $\{m_{init}\} \times V$-wise positional equilibrium. It is easy to see that a pair $(\mathcal{M} \times \mathcal{A}, f)$, where

$$f\colon M \times V \to M, \qquad f((m, v)) = m,$$

is an $\mathcal{M}$-trivial pair, by definition of $\mathcal{M} \times \mathcal{A}$. Observe that $\{m_{init}\} \times V = f^{-1}(m_{init})$, so we only have to show that $\mathcal{M} \times \mathcal{A}$ has an $f^{-1}(m_{init})$-wise positional equilibrium. Since $\mathcal{M} \times \mathcal{A}$ has at most $|\mathcal{M}| \cdot n$ nodes, it remains to explain why the assumption of Lemma 20 holds for $N = |\mathcal{M}| \cdot n$.

By the assumption of Theorem 12 all one-player arenas with at most $2|\mathcal{M}| \cdot n - 1 = 2N - 1$ nodes have a uniform $\mathcal{M}$-strategy equilibrium. In particular, this applies to any one-player arena $\mathcal{A}'$ with at most $2N - 1$ nodes belonging to some $\mathcal{M}$-trivial pair $(\mathcal{A}', f)$. By Observation 19 this means that all such $\mathcal{A}'$ have a $f^{-1}(m_{init})$-wise positional equilibrium, as required. ◀

Proof of Lemma 20. We use the same technique and terminology as in Section 4. We are now proving by induction on m the following claim: for every m and for every $\mathcal{M}$-trivial pair $(\mathcal{A}, f)$ such that $\mathcal{A}$ has m edges and at most N nodes there exists an $f^{-1}(m_{init})$-wise positional equilibrium in $\mathcal{A}$.

Induction base ($m = 1$) again requires no argument, and we proceed to the induction step. Consider any $\mathcal{M}$-trivial pair $(\mathcal{A}, f)$, where $\mathcal{A} = \langle V, V_{\text{Max}}, V_{\text{Min}}, E, \mathsf{source}, \mathsf{target}, \mathsf{col}\rangle$ has at most N nodes. Our goal is to show that $\mathcal{A}$ has an $f^{-1}(m_{init})$-wise positional equilibrium, provided that an analogous claim is already proved for all $\mathcal{M}$-trivial pairs $(\mathcal{A}', f')$ in which $\mathcal{A}'$ has at most N nodes and fewer edges than $\mathcal{A}$. Since the set of $f^{-1}(m_{init})$-wise equilibria is a Cartesian product by Lemma 15, it is enough to establish the following two claims:

(a) in $\mathcal{A}$ there exists an $f^{-1}(m_{init})$-wise equilibrium including a positional strategy of Max;

(b) in $\mathcal{A}$ there exists an $f^{-1}(m_{init})$-wise equilibrium including a positional strategy of Min.

We only show **(a)**, a proof of **(b)** is similar. As before, we may assume that $\mathcal{A}$ is not one-player so that there exists a node $w \in V_{\text{Max}}$ with out-degree at least 2. We partition the set of its out-going edges into two disjoint non-empty sets $E_1(w)$ and $E_2(w)$. Then we define arenas $\mathcal{A}_1$ and $\mathcal{A}_2$ exactly as in Section 4. Since $(\mathcal{A}, f)$ is an $\mathcal{M}$-trivial pair, then so are pairs $(\mathcal{A}_1, f)$ and $(\mathcal{A}_2, f)$. Indeed, $\mathcal{A}_1$ and $\mathcal{A}_2$ were obtained by simply throwing away some edges of $\mathcal{A}$. The remaining edges satisfy the definition of $\mathcal{M}$-triviality with respect to f just because they do so inside $\mathcal{A}$.

Note that $\mathcal{A}_1$ and $\mathcal{A}_2$ both have fewer edges than $\mathcal{A}$ and at most as many nodes. So by the induction hypothesis both these arenas have an $f^{-1}(m_{init})$-wise positional equilibrium. Let (σ_1, τ_1) be an $f^{-1}(m_{init})$-wise positional equilibrium in $\mathcal{A}_1$ and (σ_2, τ_2) be an $f^{-1}(m_{init})$-wise positional equilibrium in $\mathcal{A}_2$. Next, we define two auxiliary strategies τ_{12} and τ_{21} of Min exactly as in Section 4. Our goal is to show that either (σ_1, τ_{12}) is an $f^{-1}(m_{init})$-wise equilibrium in $\mathcal{A}$ or (σ_2, τ_{21}) is an $f^{-1}(m_{init})$-wise equilibrium in $\mathcal{A}$.

By the same argument as in Section 4, we have that τ_{12} is an $f^{-1}(m_{init})$-wise optimal response to σ_1 and τ_{21} is an $f^{-1}(m_{init})$-wise optimal response to σ_2. The main challenge is to show the opposite for at least one of the pairs (σ_1, τ_{12}) and (σ_2, τ_{21}).

For that we define a one-player arena $\mathcal{B}$ exactly as in Section 4 (see Figure 1). It has $2|V| - 1 \leq 2N - 1$ nodes. We will apply the assumption of Lemma 20 to $\mathcal{B}$. More precisely, this will be done for some $\mathcal{M}$-trivial pair which includes $\mathcal{B}$. For that we define the following mapping g from the set of nodes of $\mathcal{B}$ to the set of states of $\mathcal{M}$. Namely, if v' is a node of $\mathcal{B}$, we set $g(v') = f(v)$, where v is the prototype of v'. Observe that $(\mathcal{B}, g)$ is an $\mathcal{M}$-trivial pair. Indeed any edge of $\mathcal{B}$ is between two nodes whose prototypes are connected in $\mathcal{A}$ by an edge of the same color. Thus, by the assumption of Lemma 20, the arena $\mathcal{B}$ has a $g^{-1}(m_{init})$-wise positional equilibrium $(\widehat{\Sigma}, T)$ (as before, in $\mathcal{B}$ there are no strategies of Min other than T). It is sufficient to establish the following two claims:

- if $\widehat{\Sigma}$ applies an edge from $E_1(w)$ in w, then σ_1 is an $f^{-1}(m_{init})$-wise optimal response to τ_{12} in $\mathcal{A}$;
- if $\widehat{\Sigma}$ applies an edge from $E_2(w)$ in w, then σ_2 is an $f^{-1}(m_{init})$-wise optimal response to τ_{21} in $\mathcal{A}$.

A key observation here is that $g^{-1}(m_{init})$ is the union of the left copies of the nodes from $f^{-1}(m_{init})$ and the right copies of the nodes of $f^{-1}(m_{init})$. In fact, for a proof of the first claim we only need a fact that $g^{-1}(m_{init})$ includes all the left copies of the nodes from $f^{-1}(m_{init})$. Correspondingly, only the right copies of $f^{-1}(m_{init})$ are relevant for a proof of the second claim.

We only show the first claim, the second one can be proved similarly. As in Section 4, the argument is carried out through a positional strategy $\widehat{\sigma}$ of Max in $\mathcal{A}$ obtained by restricting $\widehat{\Sigma}$ to the left ellipse. First we observe that in any node from $f^{-1}(m_{init})$ the strategy σ_1 is at least as good against τ_{12} as $\widehat{\sigma}$. Indeed, both σ_1 and $\widehat{\sigma}$ are strategies in $\mathcal{A}_1$ whereas σ_1 is an $f^{-1}(m_{init})$-wise optimal response to τ_1 in $\mathcal{A}_1$ by definition. On the other hand, τ_{12} plays against σ_1 and $\widehat{\sigma}$ exactly as τ_1.

It remains to show that $\widehat{\sigma}$ is an optimal response to τ_{12} in any node from $f^{-1}(m_{init})$. This can be done by exactly the same argument as in the last paragraph of Section 4. A difference is that now we have a weaker assumption about $\widehat{\Sigma}$; namely, we only know that $\widehat{\Sigma}$ is optimal in the nodes from $g^{-1}(m_{init})$ (while before it was optimal everywhere in $\mathcal{B}$). Correspondingly, we are proving a weaker statement. Namely, instead of proving that $\widehat{\sigma}$ is an optimal response to τ_{12} everywhere in $\mathcal{A}$, we are only proving this for all $v \in f^{-1}(m_{init})$. It can be checked that in the argument for a specific v we only require optimality of $\widehat{\Sigma}$ in the left copy of v; so if $v \in f^{-1}(m_{init})$, then its left copy is in $g^{-1}(m_{init})$ so that the argument still works. ◀

If VNP Is Hard, Then so Are Equations for It

Mrinal Kumar
Indian Institute of Technology Bombay, India

C. Ramya
Chennai Mathematical Institute, India

Ramprasad Saptharishi
School of Technology and Computer Science, Tata Institute of Fundamental Research,
Mumbai, India

Anamay Tengse
Department of Computer Science, University of Haifa, Israel

Abstract

Assuming that the Permanent polynomial requires algebraic circuits of exponential size, we show that the class VNP *does not* have efficiently computable equations. In other words, any nonzero polynomial that vanishes on the coefficient vectors of all polynomials in the class VNP requires algebraic circuits of super-polynomial size.

In a recent work of Chatterjee, Kumar, Ramya, Saptharishi and Tengse (FOCS 2020), it was shown that the subclasses of VP and VNP consisting of polynomials with bounded integer coefficients *do* have equations with small algebraic circuits. Their work left open the possibility that these results could perhaps be extended to all of VP or VNP. The results in this paper show that assuming the hardness of Permanent, at least for VNP, allowing polynomials with large coefficients does indeed incur a significant blow up in the circuit complexity of equations.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Computational Complexity, Algebraic Circuits, Algebraic Natural Proofs

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.44

Related Version *Full Version*: `https://arxiv.org/abs/2012.07056`

Funding *C. Ramya*: Research supported by INSPIRE Faculty Fellowship of DST and by a grant from the Infosys Foundation. Part of this work was done when at the Tata Institute of Fundamental Research, Mumbai, India (DAE project 12-R&D-TFR-5.01-0500).
Ramprasad Saptharishi: Research was supported by the Department of Atomic Energy, Government of India, under project no. 12-R&D-TFR-5.01-0500, and also the SERB Ramanujan Fellowship.
Anamay Tengse: Research supported by the Israel Science Foundation (grant No. 716/20). Part of this work was done when at the Tata Institute of Fundamental Research, Mumbai, India, as a student (DAE project no. 12-R&D-TFR-5.01-0500), and later as a visitor (Prof. Prahladh Harsha's Swarnajaynti fellowship and Prof. Arkadev Chattopadhyay's Microsoft Research funds).

Acknowledgements We thank anonymous reviewers of an earlier version of this paper and Joshua Grochow, whose questions pointed us in the direction of this result. We also thank the reviewers of STACS 2022 for their helpful comments and suggestions.
We also thank Prerona Chatterjee and Ben Lee Volk for helpful discussions at various stages of this work.

1 Introduction

In the context of proving lower bounds in complexity theory, many of the existing approaches for proving Boolean circuit lower bounds were unified by Razborov and Rudich under the *Natural Proofs* framework [15] and they showed that, under standard cryptographic assumptions, any technique that fits into this framework cannot yield very strong lower

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 44; pp. 44:1–44:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

bounds. In the last few years there has been some work (e.g. [9], [10, 7]) aimed at developing an analogue of the Natural Proofs framework for algebraic circuit lower bounds. A crucial notion in this context is that of an *equation* for a class of polynomials which we now define.

For a class $\mathcal{C}$ of polynomials, an *equation* for $\mathbf{C}$ is a family of nonzero polynomials such that it vanishes on the coefficient vector of polynomials in $\mathcal{C}$.[1] Informally, an *algebraic natural proof* for a class $\mathcal{C}$ is a family of *equations* for $\mathcal{C}$ which can be computed by algebraic circuits of size and degree polynomially bounded in their number of variables. Thus, a lower bound for $\mathcal{C}$ can be proved by exhibiting an explicit polynomial on which an equation for $\mathcal{C}$ does not vanish.

Many of the known algebraic circuit lower bounds fit into this framework of algebraically natural proofs as observed by several authors [1, 9, 7, 10], thereby motivating the question of understanding whether techniques in this framework can yield strong algebraic circuit lower bounds; in particular, whether such techniques are sufficient to separate VNP from VP. In a recent breakthrough, Limaye, Srikanth and Tavenas [13] proved super-polynomial lower bounds against constant depth arithmetic circuits over fields of characteristic zero or large. It is not difficult to observe that this lower bound fits well into the natural proofs framework as it uses a complexity measure that is based on the rank of certain matrices to prove lower bounds. Thus, in the natural proofs framework, the first step towards a lower bound for VP is to understand whether VP has a family of equations which itself is in VP, that is its degree and its algebraic circuit size are polynomially bounded in the number of the variables. The next step, of course, would be to show the existence of a polynomial family in VNP which *does not* satisfy this family of equations. This work is motivated by the first step of this framework, that is the question of understanding whether natural and seemingly rich circuit classes like VP and VNP can have efficiently constructible equations. We briefly discuss prior work on this problem, before describing our results.

1.1 Complexity of Equations for classes of polynomials

In one of the first results on this problem, Forbes, Shpilka and Volk [7] and Grochow, Kumar, Saks and Saraf [10] observe that the class VP does not have efficiently constructible equations if we were to believe that there are hitting set generators for algebraic circuits with sufficiently succinct descriptions. However, unlike the results of Razborov and Rudich [15], the plausibility of the pseudorandomness assumption in [7, 10] is not very well understood. The question of understanding the complexity of equations for VP, or in general any natural class of algebraic circuits, continues to remain open.

In a recent work of Chatterjee, Kumar, Ramya, Saptharishi and Tengse [4], it was shown that if we focus on the subclass of VP (in fact, even VNP) consisting of polynomial families with bounded integer coefficients, then we indeed have efficiently computable equations. More formally, the main result in [4] was the following.

▶ **Theorem 1** ([4]). *For every constant $c > 0$, there is a polynomial family $\{P_{N,c}\} \in \mathsf{VP}_{\mathbb{Q}}$* [2] *such that for all large n and $N = \binom{n+n^c}{n}$, the following are true.*

- *For every family $\{f_n\} \in \mathsf{VNP}_{\mathbb{Q}}$, where f_n is an n-variate polynomial of degree at most n^c and coefficients in $\{-1, 0, 1\}$, we have*

$$P_{N,c}(\overrightarrow{\mathrm{coeff}}(f_n)) = 0\,.$$

[1] Strictly speaking, these notions need us to work with families of polynomials, even though we sometimes drop the word *family* for ease of exposition.

[2] For a field $\mathbb{F}$, $\mathsf{VP}_{\mathbb{F}}$ denotes the class VP where the coefficients of the polynomials are from the field $\mathbb{F}$. Similarly, $\mathsf{VNP}_{\mathbb{F}}$ denotes the class VNP where the coefficients of the polynomials are from the field $\mathbb{F}$.

- *There exists a family $\{h_n\}$ of n-variate polynomials and degree at most n^c with coefficients in $\{-1, 0, 1\}$ such that*

$$P_{N,c}(\overrightarrow{\mathrm{coeff}}(h_n)) \neq 0\,.$$

Here, $\overrightarrow{\mathrm{coeff}}(f)$ denotes the coefficient vector of a polynomial f.

Many of the natural and well studied polynomial families like the Determinant, the Permanent, Iterated Matrix Multiplication, etc., have this property of bounded coefficients, and in fact the above result even holds when the coefficients are as large as $\mathrm{poly}(N)$. Thus, Theorem 1 could be interpreted as some evidence that perhaps we could still hope to prove lower bounds for one of these polynomial families via proofs which are algebraically natural. Extending Theorem 1 to obtain efficiently constructible equations for all of VP (or even for slightly weaker models like formulas or constant depth algebraic circuits) is an extremely interesting open question. In fact, even a conditional resolution of this problem in either direction, be it showing that the bounded coefficients condition in Theorem 1 can be removed, or showing that there are no such equations, would be extremely interesting and would provide much needed insight into whether or not there *is* a natural-proofs-like barrier for algebraic circuit lower bounds.

1.2 Our results

In this paper, we show that assuming the Permanent is hard, the constraint of bounded coefficients in Theorem 1 is necessary for efficient equations for VNP. More formally, we show the following theorem.

▶ **Theorem 2** (Conditional Hardness of Equations for VNP). *Let $\varepsilon > 0$ be a constant. Suppose, for an m large enough, we have that Perm_m requires circuits of size 2^{m^ε}.*

Then, for $n = m^{\varepsilon/4}$, any $d \leq n$ and $N = \binom{n+d}{n}$, we have that every nonzero polynomial $P(x_1, \ldots, x_N)$ that vanishes on all coefficient vectors of polynomials in $\mathsf{VNP}_{\mathbb{C}}(n, d)$ has size at least $2^{0.1n^4 - 3n}$.

▶ Remark. Our proof of the above theorem easily extends to any field of characteristic zero. We shall just work with the complex numbers for better readability.

Extending the result in Theorem 2 to hardness of equations for VP, even under the assumption that Permanent is sufficiently hard, is an extremely interesting open question. Such an extension would answer the main question investigated in [7, 10] and show a natural-proofs-like barrier for a fairly general family of lower bound proof techniques in algebraic complexity. Our proof of Theorem 2 however crucially relies on some of the properties of VNP and does not appear to extend to VP.

Although the proof of the above theorem is quite elementary, the main message (in our opinion) is that we do not[3] have compelling evidence to rule out, or accept, the efficacy of algebraic natural proofs towards proving strong lower bounds for rich classes of algebraic circuits.

1.3 An overview of the proof

As was observed in [7, 10], a lower bound for equations for a class of polynomials is equivalent to showing the existence of succinctly describable hitting sets for this class. For our proof we show that, assuming that the permanent is sufficiently hard, the coefficient

[3] Or rather, the results of [4] and the above theorem seem to provide *some* evidence for both sides!

vectors of polynomials in VNP form a *hitting set* for the class VP. The connection between hardness and randomness in algebraic complexity is well known via a result of Kabanets and Impagliazzo [11], and we use this connection, along with some additional ideas for our proof. We briefly describe a high level sketch of our proof in a bit more detail now.

Kabanets and Impagliazzo [11] showed that using any explicit polynomial family $\{f_n\}$ that is sufficiently hard, one can construct a hitting set generator for VP, that is, we can construct a polynomial map $\mathsf{Gen}_f : \mathbb{F}^k \to \mathbb{F}^t$ that "fools" any small algebraic circuit C on t variables in the sense that $C(y_1, y_2, \ldots, y_t)$ is nonzero if and only if the k-variate polynomial $C \circ \mathsf{Gen}_f$ is nonzero. In a typical invocation of this result, the parameter k is much smaller than t (typically $k = \operatorname{poly}\log t$). Thus, this gives a reduction from the question of polynomial identity testing for t-variate polynomials to polynomial identity testing for k-variate polynomials. Another related way of interpreting this connection is that if $\{f_n\}$ is sufficiently hard then Gen_f is a polynomial map whose image does not have an equation with small circuit size. Thus, assuming the hardness of the Permanent, this immediately gives us a polynomial map (with appropriate parameters) such that its image does not have an efficiently constructible equation.

For the proof of Theorem 2, we show that the points in the image of the map $\mathsf{Gen}_{\mathrm{Perm}}$, can be viewed as the coefficient vectors of polynomials in VNP, or, equivalently in the terminology in [7, 10], that the Kabanets-Impagliazzo hitting set generator is VNP-succinct. To this end, we work with a specific instantiation of the construction of the Kabanets-Impagliazzo generator where the underlying construction of combinatorial designs is based on Reed-Solomon codes. Although this is perhaps the most well known construction of combinatorial designs, there are other (and in some parameters, better) constructions known. However, our proof relies on the properties of this particular construction to obtain the succinct description. Our final proof is fairly short and elementary, and is based on extremely simple algebraic ideas and making generous use of the fact that we are trying to prove a lower bound for equations for VNP and not VP.

Proof Idea

Let us assume that for some constant $\varepsilon > 0$ and for all[4] $m \in \mathbb{N}$, Perm_m requires circuits of size 2^{m^ε}. Kabanets and Impagliazzo [11] showed that, for every combinatorial design $\mathcal{D}$ (a collection of subsets of a universe with small pairwise intersection) of appropriate parameters, the map

$$\mathsf{Gen}_{\mathrm{Perm}}(\mathbf{z}) = (\mathrm{Perm}(\mathbf{z}_S) \ : \ S \in \mathcal{D})$$

where $\mathbf{z}_S$ denotes the variables of in $\mathbf{z}$ restricted to the indices in S, is a hitting set generator for circuits of size $2^{o(m^\varepsilon)}$. Our main goal is to construct a polynomial $F(\mathbf{y}, \mathbf{z})$ in VNP such that

$$F(\mathbf{y}, \mathbf{z}) = \sum_{S \in \mathcal{D}} \mathrm{mon}_S(\mathbf{y}) \cdot \mathsf{Perm}(\mathbf{z}_S) \tag{1.1}$$

By choosing parameters carefully, this would immediately imply that any equation on N-variables, for $N = \binom{n+d}{d}$, that vanishes on the coefficient vector of polynomials in $\mathsf{VNP}(n, d)$ (which are n-variate polynomials in VNP of degree at most d) requires size super-polynomial in N.

[4] To be more precise, we should work with this condition for "infinitely often" $m \in \mathbb{N}$ and obtain that VNP does not have efficient equations infinitely often. We avoid this technicality for the sake of simplicity and the proof continues to hold for the more precise version with suitable additional care.

To show that the polynomial $F(\mathbf{y}, \mathbf{z})$ in Equation 1.1 is in VNP, we use a specific combinatorial design. For the combinatorial design $\mathcal{D}$ obtained via Reed-Solomon codes, every set in the design can be interpreted as a univariate polynomial g of appropriate degree over a finite field. The degree of g (say δ) and size of the finite field (say p) are related to the parameters of the design $\mathcal{D}$. Now,

$$F(\mathbf{y}, \mathbf{z}) = \sum_{\substack{g \in \mathbb{F}_p[v] \\ \deg(g) \leq \delta}} \left(\prod_{i=0}^{\delta} y_i^{g_i} \right) \cdot \mathrm{Perm}(\mathbf{z}_{S(g)}), \tag{1.2}$$

where $(g_0, \ldots, g_\delta)$ is the coefficient vector of the univariate polynomial g. Expressing $F(\mathbf{y}, \mathbf{z})$ in Equation 1.2 as a polynomial in VNP requires us to implement the product $\left(\prod_{i=0}^{\delta} y_i^{g_i} \right)$ as a polynomial when given the binary representation of coefficients $g_0, \ldots, g_\delta$ via a binary vector $\mathbf{t}$ of appropriate length (say r). This is done via the polynomial $\mathrm{Mon}(\mathbf{t}, \mathbf{y})$ in Subsection 3.1 in a straightforward manner. Furthermore, we want to algebraically implement the selection $\mathbf{z}_S$ for a set S in the combinatorial design when given the polynomial $\mathbf{g}$ corresponding to S. This is implemented via the polynomial $\text{RS-Design}(\mathbf{t}, \mathbf{z})$ in Subsection 3.2. Finally, we have

$$F(\mathbf{y}, \mathbf{z}) = \sum_{\mathbf{t} \in \{0,1\}^r} \mathrm{Mon}(\mathbf{t}, \mathbf{y}) \cdot \mathrm{Perm}(\text{RS-Design}(\mathbf{t}, \mathbf{z}))$$

which is clearly in VNP as Perm_p is in VNP and polynomials $\mathrm{Mon}(\mathbf{t}, \mathbf{y})$ and $\text{RS-Design}(\mathbf{t}, \mathbf{z})$ are efficiently computable. We refer the reader to Section 3 for complete details.

Related results

The concept of algebraically natural proofs was first studied in the works of Forbes, Shpilka and Volk [7] and Grochow, Kumar, Saks and Saraf [10] who showed that constructing efficient equations for a class directly contradicts a corresponding *succinct* derandomization of the polynomial identity testing problem. In fact, Forbes, Shpilka and Volk [7] unconditionally ruled out equations for depth-three multilinear formulas computable by certain structured classes of algebraic circuits using this connection. However, this does not imply anything about complexity of equations for general classes of algebraic circuits such as VP and VNP. In the context of proving algebraic circuit lower bounds, Efremenko, Garg, Oliveira and Wigderson [6] and Garg, Makam, Oliveira and Wigderson [8] explore limitations of proving algebraic circuit lower bounds via rank based methods. However, these results are not directly concerned with the complexity of equations for circuit classes.

Recently, Bläser, Ikenmeyer, Jindal and Lysikov [2] studied the complexity of equations in a slightly different context. They studied a problem called "matrix completion rank", a measure for tensors that is NP-hard to compute. Assuming $\mathsf{coNP} \not\subseteq \exists\mathsf{BPP}$, they construct an explicit tensor of large (border) completion rank such that any efficient equation for the class of tensors of small completion rank must necessarily also vanish on this tensor of large completion rank. That is, efficient equations cannot certify that this specific tensor has large (border) completion rank. Subsequently, this result was generalized to *min-rank* or *slice-rank* [3]. The set-up in these papers is different from the that in our paper, and that of [10, 7]. One way to interpret this difference is that [2] shows that "variety of small completion rank tensors" cannot be "cut out" by efficient equations, whereas the set-up of [10, 7] and our paper would ask if *every* equation for this variety requires large complexity.

In the context of equations for varieties in algebraic complexity, Kumar and Volk [12] proved polynomial degree bounds on the equations of the Zariski closure of the set of non-rigid matrices as well as small linear circuits over all large enough fields.

2 Preliminaries

2.1 Notation

- We use $[n]$ to denote the set $\{1, \ldots, n\}$ and $[\![n]\!]$ to denote the set $\{0, 1, \ldots, n\}$. We also use $\mathbb{N}_{\geq 0}$ to denote the set of non-negative integers.
- We use boldface letters such as $\mathbf{x}, \mathbf{y}$ to denote tuples, typically of variables. When necessary, we adorn them with a subscript such as $\mathbf{y}_{[n]}$ to denote the length of the tuple.
- We also use $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $\prod x_i^{e_i}$. We write $\mathbf{x}^{\leq d}$ for the set of all monomials of degree at most d in $\mathbf{x}$, and $\mathbb{F}[\mathbf{x}]^{\leq d}$ for the set of polynomials in $\mathbf{x}$ over the field $\mathbb{F}$ of degree at most d.
- As usual, we identify the elements of $\mathbb{F}_p$ with $\{0, 1, \ldots, p-1\}$ and think of $[\![n]\!]$ as a subset of $\mathbb{F}_p$ in the natural way for any $n < p$.

2.2 Some basic definitions

Circuit classes

▶ **Definition 3** (Algebraic circuits). *An* algebraic circuit *is specified by a directed acyclic graph, with leaves (indegree zero; also called* inputs*) labelled by field constants or variables, and internal nodes labelled by $+$ or $\times$. The nodes with outdegree zero are called the* outputs *of the circuit. Computation proceeds in the natural way, where inductively each $+$ gate computes the sum of its children and each $\times$ gate computes the product of its children.*

The size *of the circuit is defined as the number of nodes in the underlying graph.*

▶ **Definition 4** (VP and VNP). *A family of polynomials $\{f_n\}$, where f_n is n-variate, is said to be in* VP *if* $\deg(f_n)$ *and the algebraic circuit complexity of f_n are bounded by a polynomial function of n. That is, there is a constant $c \geq 0$ such that for all large enough n we have* $\deg(f_n), \mathrm{size}(f_n) \leq n^c$.

A family of polynomials $\{f_n\}$ is said to be in VNP *if there is a family $\{g_n(\mathbf{x}_{[n]}, \mathbf{y}_{[m]})\} \in$ VP such that m is bounded by a polynomial function of n and*

$$f_n(\mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^m} g_n(\mathbf{x}, \mathbf{y}).$$

For some $n, d \in \mathbb{N}$, let $\mathcal{C}_{n,d}$ be a class of n-variate polynomials of *total* degree at most d. That is, $\mathcal{C}_{n,d} \subseteq \mathbb{F}[\mathbf{x}]^{\leq d}$. Similarly, we will use $\mathsf{VP}(n, d)$ and $\mathsf{VNP}(n, d)$ to denote the intersection of VP and VNP respectively, with $\mathbb{F}[\mathbf{x}_{[n]}]^{\leq d}$.

Equations and succinct hitting sets

▶ **Definition 5** (Equations for a class). *For $N = \binom{n+d}{n}$, a nonzero polynomial $P_N(\mathbf{Z})$ is called an* equation *for $\mathcal{C}_{n,d}$ if for all $f(\mathbf{x}) \in \mathcal{C}_{n,d}$, we have that $P_N(\overrightarrow{\mathrm{coeff}}(f)) = 0$, where $\overrightarrow{\mathrm{coeff}}(f)$ is the coefficient vector of f.*

Alternatively, we also say that a polynomial $P_N(\mathbf{Z})$ *vanishes* on the coefficient vectors of polynomials in class $\mathcal{C}$ if $P_N(\overrightarrow{\mathrm{coeff}}(f)) = 0$ for all $f \in \mathcal{C}$.

▶ **Definition 6** (Hitting Set Generator (HSG))**.** *A polynomial map $G : \mathbb{F}^\ell \to \mathbb{F}^n$ given by $G(z_1, \ldots, z_\ell) = (g_1(\mathbf{z}), \ldots, g_n(\mathbf{z}))$ is said to be a* hitting set generator (HSG) *for a class $\mathcal{C} \subseteq \mathbb{F}[\mathbf{x}]$ of polynomials if for all nonzero $P \in \mathcal{C}$, $P \circ G = P(g_1, \ldots, g_n) \not\equiv 0$.*

We review the definition of *succinct hitting sets* introduced [10, 7].

▶ **Definition 7** (Succinct Hitting Sets for a class of polynomials [10, 7])**.** *For $N = \binom{n+d}{n}$, we say that a class of N-variate polynomials $\mathcal{D}_N$* has $\mathcal{C}_{n,d}$-succinct hitting sets *if for all nonzero $P_N(\mathbf{Z}) \in \mathcal{D}_N$, there exists some $f \in \mathcal{C}_{n,d}$ such that $P_N(\overrightarrow{\mathrm{coeff}}(f)) \neq 0$.*

Hardness to randomness connection

For our proofs, we will need the following notion of combinatorial designs, which is a collection of subsets of a universe with small pairwise intersection.

▶ **Definition 8** (Combinatorial designs)**.** *A family of sets $\{S_1, \ldots, S_N\} \subseteq [\ell]$ is said to be an (ℓ, m, n)-design if*

- *$|S_i| = m$ for each $i \in [N]$*
- *$|S_i \cap S_j| < n$ for any $i \neq j$.*

Kabanets and Impagliazzo [11] obtain hitting set generators from polynomials that are hard to compute for algebraic circuits. The following lemma is crucial to the proof of our main theorem.

▶ **Lemma 2.1** (HSG from Hardness [11])**.** *Let $\{S_1, \ldots, S_N\}$ be an (ℓ, m, n)-design and $f(\mathbf{x}_m)$ be an m-variate, individual degree d polynomial that requires circuits of size s. Then for fresh variables $\mathbf{y}_\ell$, the polynomial map $\text{KI-gen}_{(N,\ell,m,n)}(f) : \mathbb{F}^\ell \to \mathbb{F}^N$ given by*

$$(f(\mathbf{y}_{S_1}), \ldots, f(\mathbf{y}_{S_N})) \tag{2.2}$$

is a hitting set generator for all circuits of size at most $\left(\frac{s^{0.1}}{N(d+1)^n}\right)$.

3 Proof of the main theorem

Notation

1. For a vector $\mathbf{t} = (t_1, \ldots, t_r)$, we will use the short-hand $t^{(a)}_{i,j}$ to denote the variable $t_{(i \cdot a + j + 1)}$. This would be convenient when we consider the coordinates of $\mathbf{t}$ as blocks of length a.
2. For integers a, p, we shall use $\mathrm{Mod}(a, p)$ to denote the unique integer $a_p \in [0, p-1]$ such that $a_p = a \bmod p$.

As mentioned in the overview, the strategy is to convert the hitting set generator given in (2.2) into a succinct hitting set generator. Therefore, we would like to associate the coordinates of (2.2) into coefficients of a suitable polynomial. That is, we would like to build a polynomial in VNP of the form

$$g(y_1, \ldots, y_\ell, z_1, \ldots, z_t) = \sum_{m \in \mathbf{y}^{\leq d}} m \cdot f(\mathbf{z}_{S_m})$$

with the monomials $m \in \mathbf{y}^{\leq d}$ suitably indexing into the sets of the combinatorial design. The above expression already resembles a VNP-definition and with a little care this can be made effective. We will first show that the different components of the above expression can be made succinct using the following constructions.

3.1 Building monomials from exponent vectors

For $n, r \in \mathbb{N}$, let $a = \lfloor r/n \rfloor$, and define $\mathrm{Mon}_{r,n}(\mathbf{t}, \mathbf{y})$ as follows.

$$\mathrm{Mon}_{r,n}(t_1, \ldots, t_r, y_1, \ldots, y_n) = \prod_{i=0}^{n-1} \prod_{j=0}^{a-1} \left(t_{i,j}^{(a)} y_{i+1}^{2^j} + (1 - t_{i,j}^{(a)}) \right)$$

The following observation is now immediate from the definition above.

▶ **Observation 9.** *For any $(e_1, \ldots, e_n) \in [\![d]\!]^n$, we have*

$$\mathrm{Mon}_{r,n}(\mathrm{Bin}(e_1), \ldots, \mathrm{Bin}(e_n), y_1, \ldots, y_n) = y_1^{e_1} \cdots y_n^{e_n},$$

where $\mathrm{Bin}(e)$ is the tuple corresponding to the binary representation of e, and $r = n \cdot \lceil \log_2 (d+1) \rceil$. Furthermore, the polynomial $\mathrm{Mon}_{r,n}$ is computable by an algebraic circuit of size $\mathrm{poly}(n, r)$.

3.2 Indexing Combinatorial Designs Algebraically

Next, we need to effectively compute the hard polynomial f on sets of variables in a combinatorial design, indexed by the respective monomials. We will need to simulate some computations modulo a fixed prime p. The following claim will be helpful for that purpose.

▷ Claim 10. For any $i, b, p \in \mathbb{N}_{\geq 0}$ with $i \leq p$, there exists a unique univariate polynomial $Q_{i,b,p}(v) \in \mathbb{Q}[v]$ of degree at most b such that

$$Q_{i,b,p}(a) = \begin{cases} 1 & \text{if } 0 \leq a < b \text{ and } a \equiv i \pmod{p}, \\ 0 & \text{if } 0 \leq a < b \text{ and } a \not\equiv i \pmod{p}. \end{cases}$$

Proof. We can define a unique univariate polynomial $Q_{i,b,p}(v)$ satisfying the conditions of the claim via interpolation to make a unique univariate polynomial take a value of 0 or 1 according to the conditions of the claim. Since, there are b conditions, there always exists such a polynomial of degree at most b. ◁

For any $n, b, p \in \mathbb{N}_{\geq 0}$ with $n \leq p$, define

$$\mathrm{Sel}_{n,b,p}(u_1, \ldots, u_n, v) \triangleq \sum_{i=1}^{n} u_i \cdot Q_{i,b,p}(v).$$

▶ **Observation 11.** *For any $n, b, p \in \mathbb{N}_{\geq 0}$ with $n \leq p$, for any $0 \leq a < b$, we have that*

$$\mathrm{Sel}_{n,b,p}(u_1, \ldots, u_n, a) = u_{\mathrm{Mod}(a,p)} = u_{a \bmod p}$$

The degree of $\mathrm{Sel}_{n,b,p}$ is at most $(b+1)$ and can be computed by an algebraic circuit of size $\mathrm{poly}(b)$.

Proof. From the definition of the univariate polynomial $Q_{i,b,p}(v)$ of degree b in Claim 10, $Q_{i,b,p}(a)$ outputs 1 if and only if $i = a \bmod p$. Hence, $\mathrm{Sel}_{n,b,p}(u_1, \ldots, u_n, a)$ is $u_{a \bmod p}$ and is of degree at most $(b+1)$. ◀

And finally, we choose a specific combinatorial design to instantiate Lemma 2.1.

3.3 Reed-Solomon-based combinatorial designs

For any prime p and any choice of $a \leq p$, the following is an explicit construction of a (p^2, p, a)-combinatorial design of size p^a, defined as follows:

> For every univariate polynomial $g(t) \in \mathbb{F}_p[t]$ of degree less than a, we add the set $S_g = \{(i, g(i)) \ : \ i \in \mathbb{F}_p\} \subseteq \mathbb{F}_p \times \mathbb{F}_p$ to the collection.

Since any two distinct univariate polynomials of degree less than a can agree on at most a points, it follows that the above is indeed a (p^2, p, a)-design.

The advantage of this specific construction is that it can be made succinct as follows. For $r = a \cdot \lfloor \log_2 p \rfloor$, let $t_1, \ldots, t_r$ be variables taking values in $\{0, 1\}$. The values assigned to $\mathbf{t}$-variables can be interpreted as a univariate over $\mathbb{F}_p$ of degree $< a$ by considering $\mathbf{t} \in \{0,1\}^r$ as a matrix with a rows and $\lfloor \log_2 p \rfloor$ columns each [5]. The binary vector in each row represents an element in $\mathbb{F}_p$. We illustrate this with an example.

$$\mathbf{t} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \qquad \longrightarrow \qquad \begin{pmatrix} 7 \\ 2 \\ 1 \\ 4 \\ 3 \end{pmatrix} \cong g(v)$$

For $p = 11$, $a = 5$, $g(v) = 7 + 2v + v^2 + 4v^3 + 2v^4 \in \mathbb{F}_{11}[v]$,

$\mathbf{t}$ is a 5×3 matrix that encodes the coefficients of $g(v)$.

Let $\mathbf{z}$ denote the p^2 variables $\{z_1, \ldots, z_{p^2}\}$, put in into a $p \times p$ matrix. Let S be a set in the Reed-Solomon-based (p^2, p, a)-combinatorial design. We want to implement the selection $\mathbf{z}_S$ algebraically. In the following, we design a vector of polynomials that outputs the vector of variables $\left(z^{(p)}_{0, g(0) \bmod p}, \ldots, z^{(p)}_{p-1, g(p-1) \bmod p}\right)$. Note that as mentioned above the polynomial g can be specified via variables $t_1, \ldots, t_r$. That is,

$$\begin{aligned} \text{RS-Design}_{p,a}(t_1, \ldots, t_r, z_1, \ldots, z_{p^2}) &\in (\mathbb{F}[\mathbf{t}, \mathbf{z}])^p \quad , \quad \text{for } r = a \cdot \lfloor \log_2 p \rfloor, \\ \text{RS-Design}_{p,a}(t_1, \ldots, t_r, z_1, \ldots, z_{p^2})_{i+1} &= \text{Sel}_{p,p^3,p}\left(z^{(p)}_{i,0}, \ldots, z^{(p)}_{i,p-1}, R_{i,a,p}(\mathbf{t})\right), \quad \forall i \in \mathbb{F}_p, \\ \text{where } R_{i,a,p}(\mathbf{t}) &= \sum_{j=0}^{a-1} \left[\left(\sum_{k=0}^{\ell_p - 1} t^{(\ell_p)}_{j,k} \cdot 2^k \right) \cdot \text{Mod}(i^j, p) \right], \\ \text{with } \ell_p &= \lfloor \log_2 p \rfloor . \end{aligned}$$

▶ **Observation 12.** *For any prime p, $a \leq p$, and $\mathbf{t} \in \{0,1\}^r$ for $r = a \cdot \lfloor \log_2 p \rfloor$, we have*

$$\text{RS-Design}_{p,a}(\mathbf{t}, \mathbf{z}) = \left(z_{i, g(i)} \ : \ i \in \mathbb{F}_p \right),$$

where $g(v) \in \mathbb{F}_p[v]$ is the univariate whose coefficient vector is represented by the bit-vector $\mathbf{t}$. Furthermore, the polynomial RS-Design$_{p,a}$ *is computable by an algebraic circuit of size* poly(p).

[5] Working with $\lfloor \log_2 p \rfloor$ bits (as opposed to $\lceil \log_2 p \rceil$) makes the proofs much simpler, and does not affect the size of the design by much.

Proof. Fix some $\mathbf{t} \in \{0,1\}^r$. From the definition of $R_{i,a,p}(\mathbf{t})$, it is clear that $R_{i,a,p}(\mathbf{t})$ returns an integer α such that $g(i) = \alpha \bmod p$ where $\mathbf{t}$ encodes the coefficients of the polynomial $g(t)$ in binary. Furthermore, since $\mathrm{Mod}(i^j, p)$ is the unique integer $c \in [0, p-1]$ with $c = i^j \bmod p$, it also follows that $R_{i,a,p}(\mathbf{t})$ is an integer in the range $[0, p^3]$. Hence,

$$\mathrm{Sel}_{p,p^3,p}\left(z^{(p)}_{i,0}, \ldots, z^{(p)}_{i,p-1}, R_{i,a,p}(\mathbf{t})\right) = z_{i,g(i)}$$

as claimed. ◀

3.4 The VNP-Succinct-KI generator

We are now ready to show the VNP-succinctness of the Kabanets-Impagliazzo hitting set generator when using a hard polynomial from VNP and a Reed-Solomon-based combinatorial design.

For a prime p and for the largest number m such that $m^2 \leq p$, we will use $\mathrm{Perm}_{[p]} \in \mathbb{F}[\mathbf{y}_{[p]}]$ to denote Perm_m applied to the first m^2 variables of $\mathbf{y}$.

We now define the polynomial $F_{n,a,p}(\mathbf{y}_{[n]}, \mathbf{z}_{[p^2]})$ as follows.

$$F_{n,a,p}(y_1, \ldots, y_n, z_1, \ldots, z_{p^2}) = \sum_{\mathbf{t} \in \{0,1\}^r} \mathrm{Mon}_{r,n}(\mathbf{t}, \mathbf{y}) \cdot \mathrm{Perm}_{[p]}(\text{RS-Design}_{p,a}(\mathbf{t}, \mathbf{z})) \qquad (3.1)$$
$$\text{where } r = a \cdot \lfloor \log_2 p \rfloor$$

It is evident from the above definition that the polynomial $F_{n,a,p}(\mathbf{y}, \mathbf{z})$ is in VNP for any p that is $\mathrm{poly}(n)$, when seen as a polynomial in $\mathbf{y}$-variables with coefficients from $\mathbb{C}[\mathbf{z}]$.

From the construction, we have that

$$F_{n,a,p}(y_1, \ldots, y_n, z_1, \ldots z_{p^2}) = \sum_{\mathbf{e}} \mathbf{y}^{\mathbf{e}} \cdot \mathrm{Perm}_{[p]}(\mathbf{z}_{S_{\mathbf{e}}}),$$

where $\{S_{\mathbf{e}}\}$ is an appropriate ordering of the Reed-Solomon-based (p^2, p, a)-combinatorial design of size p^a, described in Subsection 3.3. Note that we define the polynomial $F_{n,a,p}(\mathbf{y}, \mathbf{z})$ with three parameters n, a, p although for our purposes we will only use $a = n$.

3.5 Putting it all together

We are now ready to show that if the Permanent polynomial is exponentially hard, then any polynomial P that vanishes on the coefficient vectors of all polynomials in the class VNP requires super-polynomial size to compute it.

▶ **Theorem 2** (Conditional Hardness of Equations for VNP). *Let $\varepsilon > 0$ be a constant. Suppose, for an m large enough, we have that* Perm_m *requires circuits of size 2^{m^ε}.*

Then, for $n = m^{\varepsilon/4}$, any $d \leq n$ and $N = \binom{n+d}{n}$, we have that every nonzero polynomial $P(x_1, \ldots, x_N)$ that vanishes on all coefficient vectors of polynomials in $\mathsf{VNP}_{\mathbb{C}}(n,d)$ *has size at least $2^{0.1n^4 - 3n}$.*

Proof. Let p be the smallest prime larger than m^2; we know that $p \leq 2m^2$. We will again use $\mathrm{Perm}_{[p]} \in \mathbb{F}[\mathbf{y}_{[p]}]$ to denote Perm_m acting on the first m^2 variables of $\mathbf{y}$. Therefore, if Perm_m requires size 2^{m^ε} then so does $\mathrm{Perm}_{[p]}$.
Consider the polynomial $F_{n,n,p}(\mathbf{y}_{[n]}, \mathbf{z}_{[p^2]}) \in \mathsf{VNP}$ defined in (3.1), which we interpret as a polynomial in $\mathbf{y}$ with coefficients in $\mathbb{C}[\mathbf{z}]$. The individual degree in $\mathbf{y}$ is at least d, and at most p.

Let $F^{\leq d}_{n,n,p}(\mathbf{y}_{[n]}, \mathbf{z}_{[p^2]})$ denote the polynomial obtained from $F_{n,n,p}$ by discarding all terms whose total degree in $\mathbf{y}$ exceeds d. By standard homogenisation arguments, it follows that $F^{\leq d}_{n,n,p} \in \mathsf{VNP}$ as well. Therefore,

$$F^{\leq d}_{n,n,p}(\mathbf{y}, \mathbf{z}) = \sum_{\deg(\mathbf{y}^{\mathbf{e}}) \leq d} \mathbf{y}^{\mathbf{e}} \cdot \mathrm{Perm}_{[p]}(\mathbf{z}_{S_{\mathbf{e}}}),$$

where $S_{\mathbf{e}}$, for various $\mathbf{e}$, is an appropriate indexing into a (p^2, p, n)-combinatorial design of size N. Since the individual degree in $\mathbf{y}$ of $F_{n,n,p}$ was at least d, every coefficient of $F^{\leq d}_{n,n,p}$ is $\mathrm{Perm}_{[p]}(\mathbf{z}_S)$ for some S in the combinatorial design. In other words, the coefficient vector of $F^{\leq d}_{n,n,p}$ is precisely $\text{KI-gen}_{N,p^2,p,n}(\mathrm{Perm}_{[p]})$.

Suppose $P(x_1, \ldots, x_N)$ is a nonzero equation for $\mathsf{VNP}(n,d)$, then in particular it should be zero on the coefficient vector of $F^{\leq d}_{n,n,p}(\mathbf{y}, \mathbf{a}) \in \mathsf{VNP}$ for any $\mathbf{a} \in \mathbb{C}^{p^2}$. By the Polynomial Identity Lemma [14, 5, 17, 16], this implies that P must be zero on the coefficient vector of $F^{\leq d}_{n,n,p}(\mathbf{y}, \mathbf{z}) \in (\mathbb{C}[\mathbf{z}])[\mathbf{y}]$, where coefficients are formal polynomials in $\mathbb{C}[\mathbf{z}]$. Since the coefficient vector of $F^{\leq d}_{n,n,p}(\mathbf{y}, \mathbf{z})$ is just $\text{KI-gen}_{N,p^2,p,n}(\mathrm{Perm}_{[p]})$, the contrapositive of Lemma 2.1 gives that

$$\begin{aligned} \mathrm{size}(P) &> \frac{\mathrm{size}(\mathrm{Perm}_{[p]})^{0.1}}{N \cdot 2^n} > \frac{\mathrm{size}(\mathrm{Perm}_m)^{0.1}}{N \cdot 2^n} \\ \implies \mathrm{size}(P) &> \frac{2^{0.1 m^{\varepsilon}}}{N \cdot 2^n} \end{aligned}$$

Since $N = \binom{n+d}{n} \leq 2^{2n}$, it follows that size$(P)$ is at least at least $2^{0.1n^4 - 3n}$. ◀

Concluding that VNP has no efficient equations

Note that for a family $\{P_N\}$ to be a *family of equations* for a class $\mathcal{C}$, we want that for *all large enough* n, the corresponding polynomial P_N should vanish on the coefficient vectors of all n-variate polynomials in $\mathcal{C}$. This condition is particularly important if we want to use equations for $\mathcal{C}$ to prove lower bounds against it, since a family of polynomials $\{f_n\}$ is said to be computable in size $s(n)$ if $\mathrm{size}(f_n) \leq s(n)$ for *all large enough* n.

Theorem 2 shows that, for m large enough, if there is a constant $\varepsilon > 0$ such that $\mathrm{size}(\mathrm{Perm}_m) \geq 2^{m^{\varepsilon}}$, then for $n = m^{\varepsilon/4}$ and any $d \leq n$, the coefficient vectors of polynomials in $\mathsf{VNP}(n,d)$ form a hitting set for all N-variate polynomials (where $N = \binom{n+d}{d}$) of degree poly(N) that are computable by circuits of size poly(N). Now suppose the Permanent family is $2^{m^{\varepsilon}}$-hard for a constant $\varepsilon > 0$, which means that Perm_m is $2^{m^{\varepsilon}}$-hard for *infinitely many* $m \in \mathbb{N}$. Then using Theorem 2, we can conclude that for any family $\{P_N\} \in \mathsf{VP}$, we must have for *infinitely many* n that $P_N(\overrightarrow{\mathrm{coeff}}(f_n)) \neq 0$ for some $f_n \in \mathsf{VNP}$, which then shows that $\{P_N\}$ is not a family of equations for VNP.

4 Discussion and Open Problems

In the context of proving circuit lower bounds, and in relation to the notion of *algebraically natural proofs*, an interesting question that emerges from the recent work of Chatterjee, Kumar, Ramya, Saptharishi, Tengse [4] (stated in Theorem 1) is whether the condition of "small coefficients" is necessary for efficiently constructible equations to exist, especially for the class VP. While this question remains open for VP, our result shows that this additional restriction on the coefficients is essentially vital for the existence of efficiently constructible equations for the class VNP, and therefore provides strong evidence *against* the existence of efficient equations for VNP.

In light of Theorem 1 and Theorem 2 for VNP, one could make a case that equations for VP might also incur a super-polynomial blow up, without the restriction on coefficients. On the other hand, it could also be argued that an analogue of Theorem 2 may not be true for VP, since our proof crucially uses the fact that VNP is "closed under exponential sums". In fact, our proof essentially algebrizes the intuition that coefficient vectors of polynomials in VNP "look random" to a polynomial in VP, *provided that* VNP was exponentially more powerful than VP.

Thus, along with the previously known results on efficient equations for polynomials in VP with bounded coefficients, our result highlights that the existence of such equations for VP in general continues to remain an intriguing mystery.

Open Problems

We now conclude with some possible directions for extending our results.

- Perhaps the most interesting question here is to prove an analogue of Theorem 2 for equations for VP. This would provide concrete evidence for the possibility that we cannot hope to prove very strong lower bounds for algebraic circuits using proofs which proceed via efficiently constructible equations, from a fairly standard complexity theoretic assumption.
- At the moment, we cannot rule out the possibility of there being efficient equations for VP in general; it may be possible that the bounded coefficients condition in Theorem 1 can be removed. In particular, the question of proving upper bounds on the complexity of equations for VP is also extremely interesting, even if one proves such upper bounds under some reasonable complexity theoretic assumptions. A first step perhaps would be to prove upper bounds on the complexity of potentially simpler models, like formulas, algebraic branching programs or constant depth circuits. From the works of Forbes, Shpilka and Volk [7], we know that such equations for structured subclasses of VP (like depth-3 multilinear circuits) cannot be *too* simple (such as sparse polynomials, depth-3 powering circuits, etc.). Can we prove a non-trivial upper bound for equations for these structured classes within VP?
- Another question of interest would be to understand if the hardness assumption in Theorem 2 can be weakened further. For instance, is it true that VNP does not have efficiently constructible equations if $\mathsf{VP} \neq \mathsf{VNP}$, or if Perm_n requires circuits of size $n^{\mathrm{poly}\log(n)}$? The current proof seems to need an exponential lower bound for the Permanent.

References

1 Scott Aaranson and Andrew Drucker. Arithmetic natural proofs theory is sought. Shtetl Optimized: Scott Aaranson's Blog, 2008. URL: https://www.scottaaronson.com/blog/?p=336.

2 Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1193–1206, 2018. doi:10.1145/3188745.3188832.

3 Markus Bläser, Christian Ikenmeyer, Vladimir Lysikov, Anurag Pandey, and Frank-Olaf Schreyer. Variety membership testing, algebraic natural proofs, and geometric complexity theory. *CoRR*, abs/1911.02534, 2019. arXiv:1911.02534.

4 Prerona Chatterjee, Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. On the existence of algebraically natural proofs. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 870–880. IEEE, 2020. arXiv:2004.14147, doi:10.1109/FOCS46700.2020.00085.

5 Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

6 Klim Efremenko, Ankit Garg, Rafael Oliveira, and Avi Wigderson. Barriers for rank methods in arithmetic complexity. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 1:1–1:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.1.

7 Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct hitting sets and barriers to proving lower bounds for algebraic circuits. *Theory of Computing*, 14(1):1–45, 2018. doi:10.4086/toc.2018.v014a018.

8 Ankit Garg, Visu Makam, Rafael Oliveira, and Avi Wigderson. More barriers for rank methods, via a "numeric to symbolic" transfer. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 824–844. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00054.

9 Joshua A. Grochow. Unifying known lower bounds via geometric complexity theory. *Computational Complexity*, 24(2):393–475, 2015. doi:10.1007/s00037-015-0103-x.

10 Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *CoRR*, abs/1701.01717, 2017. arXiv:1701.01717.

11 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.

12 Mrinal Kumar and Ben Lee Volk. A polynomial degree bound on equations for non-rigid matrices and small linear circuits. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 9:1–9:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.9.

13 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. *Electron. Colloquium Comput. Complex.*, page 81, 2021. URL: https://eccc.weizmann.ac.il/report/2021/081.

14 Øystein Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922.

15 Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

16 Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

17 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.

Determining a Slater Winner Is Complete for Parallel Access to NP

Michael Lampis

Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France

Abstract

We consider the complexity of deciding the winner of an election under the Slater rule. In this setting we are given a tournament $T = (V, A)$, where the vertices of V represent candidates and the direction of each arc indicates which of the two endpoints is preferable for the majority of voters. The *Slater score* of a vertex $v \in V$ is defined as the minimum number of arcs that need to be reversed so that T becomes acyclic and v becomes the winner. We say that v is a Slater winner in T if v has minimum Slater score in T.

Deciding if a vertex is a Slater winner in a tournament has long been known to be NP-hard. However, the best known complexity upper bound for this problem is the class Θ_2^p, which corresponds to polynomial-time Turing machines with parallel access to an NP oracle. In this paper we close this gap by showing that the problem is Θ_2^p-complete, and that this hardness applies to instances constructible by aggregating the preferences of 7 voters.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Slater winner, Feedback Arc Set, Tournaments

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.45

Related Version *Full Version*: https://arxiv.org/abs/2103.16416

Acknowledgements I am grateful to Jérôme Lang for letting me know about this problem and for correctly conjecturing that it is complete for Θ_2^p.

1 Introduction

Voting rules, which are a topic of central interest in computational social choice, are schemes which allow us to aggregate the preferences of a set of voters among a set of candidates, in order to select a single winner who is most compatible with the voters' wishes. The main challenge of this area is that, even if the preferences of each voter are internally consistent (that is, each voter has a complete ranking of all candidates), it is easy to run into situations such as the famous Condorcet paradox where collective preferences are cyclic and hence no clear winner exists. Many rules have therefore been proposed to deal with this situation and select a winner who is as acceptable as possible to as many voters as possible.

In this paper we investigate the computational complexity of a classical and very natural such voting scheme that is often referred to as the Slater rule. Intuitively, the idea of the Slater rule is the following: we consider every possible pair of candidates in our pool a, b and check whether the majority of voters prefers a or b. This allows us to construct a tournament T that depicts the results of each pariwise matchup between candidates. If T is transitive (that is, acyclic), then picking a winner is easy. If not, the Slater rule is that we should select as the winner a candidate who is the winner of a transitive tournament T' that is at minimum edit distance from T. In other words, a candidate c is a Slater winner if the number of pairwise matchups that we need to ignore to make c a clear winner is minimized.

More formally, the problem we consider is defined as follows. We are given a set V of n candidates and the preferences of m voters, where each voter's preferences are given as a total ordering of V. We determine a pair-wise relation on V as follows: for $a, b \in V$ we say that a wins against b if the majority of voters prefers candidate a over candidate b. In this way, assuming that there are no ties (which is guaranteed if the number of voters is odd), we can construct a tournament $T = (V, A)$, where we have the arc $b \rightarrow a$ (that is $(b, a) \in A$) if a wins against b. In this setting, the Slater score of a candidate c is the minimum number of arcs of T that need to be reversed so that T becomes transitive (acyclic) with c being placed last (that is, with c being a sink). The Slater winner of a tournament is a candidate with minimum Slater score. Intuitively, a candidate c is a Slater winner if there exists a linear ordering $\prec$ of the candidates that ranks c as the winner and is as compatible as possible with the voters' aggregated preferences, in the sense that the edit distance between $\prec$ and T is minimum.

The notion of Slater winner is very well-studied and can be seen as a special case of Kemeny voting. Indeed, in Kemeny voting we construct a weighted tournament where the weight of the arc $b \rightarrow a$ denotes the margin of victory of a over b. In this sense, the Slater system corresponds to a version of Kemeny voting where we only retain as information which of the two candidates would win a head-to-head match-up, but ignore the margin of victory. In other words, Slater voting is the special case of Kemeny voting where the arcs are unweighted. For more information about these and other related voting systems, we refer the reader to [5].

The main question we are interested in in this paper is the computational complexity of determining if a vertex v of a tournament is a Slater winner. It has long been known that this question is at least NP-hard [15]. Indeed, it is not hard to see that if we had an oracle for the Slater problem we would be able to produce in polynomial time an ordering of any tournament in a way that minimizes the number of inversed arcs. This would solve the Feedback Arc Set problem, which is known to be NP-complete on tournaments [1, 2, 7, 9]. On the other hand, membership of this problem in NP is not obvious. The best currently known upper bound on its complexity is the class Θ_2^p, shown by Hudry [8, 15].

The class Θ_2^p seems like a natural home for the Slater problem. As a reminder, this class captures as a model of computation Turing machines that run in polynomial time and which are allowed to use an NP oracle either a polynomial number of times non-adaptively (that is, with questions not being allowed to depend on previous answers), or a logarithmic number of times adaptively. Hence, this class is often called "Parallel Access to NP" and written as $\mathrm{P}_{||}^{\mathrm{NP}}$. Intuitively, solving the Slater problem requires us to calculate exactly a value that is NP-hard to compute (the minimum feedback arc set of a tournament). This can be done either by asking polynomially many non-adaptive NP queries to an oracle (for each $k = 1, 2, \dots$ we ask if the feedback arc set has size at most k), or a logarithmic number of adaptive queries (where we essentially perform binary search). It has therefore been conjectured that determining if a candidate is a Slater winner is not just NP-hard, but Θ_2^p-complete [4, 5, 15]. We recall that Θ_2^p is strongly suspected to be a much larger class than NP – indeed, because Θ_2^p contains all of the so-called Boolean hierarchy of classes, it is known that if it were the case that $\Theta_2^p = NP$, then the polynomial hierarchy would collapse [6]. Hence, the difference between the known upper and lower bounds on the complexity of determining a Slater winner is not trivial.

The result we present in this paper settles this problem. We confirm the conjecture that determining the Slater winner of a tournament is indeed Θ_2^p-complete. This places Slater voting in the same class as related voting schemes, such as Kemeny [14], Dodgson [12], and Young [16]. It also places it in the same class as the Slater rule used in [11] for a more general judgment aggregation problem. We prove this result by modifying the reduction of Conitzer [9], which showed that Feedback Arc Set on tournaments is NP-complete. The main difference is that, rather than reducing from SAT, we need to reduce from a Θ_2^p-complete variant, where we are looking for a maximum weight satisfying assignment that sets a certain variable to True. This forces us to significantly complicate the reduction because we need to encode in the objective function not only the number of satisfied clauses but also the weight of the corresponding assignment.

Having settled the worst-case complexity of the problem in general, we go on to consider a related question: what is the minimum number of voters for which determining a Slater winner is Θ_2^p-complete? The motivation behind this question is that, even though any tournament can be constructed by aggregating the preferences of a large enough number of voters[1], if the number of voters is limited, some tournaments can never arise. Hence the problem may conceivably be easier if the number of voters is bounded. In the case of the Slater rule, Bachmeier et al.[3] have shown that determining the Slater winner remains NP-hard for 7 voters. By reusing and slightly adjusting their arguments we improve their complexity lower bound to Θ_2^p-completeness for 7 voters.

2 Definitions and Preliminaries

A tournament is a directed graph $G = (V, A)$ such that for all $x, y \in V$, exactly one of the arcs $(x, y), (y, x)$ appears in A. A feedback arc set (fas) of a digraph $G = (V, A)$ is a set of arcs $A' \subseteq A$ such that deleting A' from G results in an acyclic digraph. If G is a tournament and A' is a fas of G, then the tournament obtained from G by reversing the direction of all arcs of A' is acyclic (or transitive). We will say that a total ordering $\prec$ of the vertices of a digraph $G = (V, A)$ *implies* the fas $S = \{(x, y) \mid (x, y) \in A,\ y \prec x\}$ (in the sense that S is the set of arcs that disagree with the ordering). We will say that an ordering of V is optimal if the fas it implies has minimum size.

Given a digraph $G = (V, A)$ and $v \in V$, we say that v is a Slater winner if for some $k \geq 0$ the following hold: (i) there exists a fas $S \subseteq A$ of G, such that v is a sink of $G - S$ and $|S| = k$ (ii) every fas of G has size at least k. If v is a Slater winner in $G = (V, A)$, then a winning ordering for v is a linear ordering of V that places v last and implies a fas of G of minimum size.

In a digraph $G = (V, E)$, a set $M \subseteq V$ is a module if the following holds: for all $x, y \in M$ and $z \notin M$ we have $(x, z) \in E \leftrightarrow (y, z) \in E$ and $(z, x) \in E \leftrightarrow (z, y) \in E$. In other words, every vertex outside M that has an arc to (respectively from) a vertex of M, has arcs to (respectively from) all of M. The following lemma, given by Conitzer [9] with slightly different terminology, states that the vertices of a module can, without loss of generality, always be ordered together. We say that the vertices of a set S are contiguous in an ordering $\prec$ if there are no $x, y \in S$, $z \notin S$ such that $x \prec z \prec y$.

▶ **Lemma 1.** *Let $G = (V, A)$ be a digraph, $v \in V$ a vertex, and suppose we have a partition of V into k non-empty modules $V = M_1 \uplus M_2 \uplus \ldots \uplus M_k$. If v is a Slater winner of G, then there exists a winning ordering for v such that for all $i \in [k]$, the vertices of M_i are contiguous.*

Proof. Suppose $k > 1$ (otherwise the claim is trivial) and consider an ordering $\prec$ that is winning for v. We will say that a set of vertices $S \subseteq V$ is a *block* of $\prec$ if (i) $S \subseteq M_i$ for some $i \in \{1, \ldots, k\}$; (ii) S is contiguous; (iii) S is maximal, that is, adding any vertex to S violates one of the two preceding properties.

[1] This is a classical result known in the literature as McGarvey's theorem.

If the number of blocks is equal to k we are done, as each block is equal to a module so we have an ordering where each module is contiguous. If we have at least $k+1$ blocks, we will explain how to edit the ordering so that it remains winning for v, it implies a fas of the same size, and the number of blocks decreases. Repeating this process until we have k blocks completes the proof.

Consider two vertices x, y, with $x \prec y$, which belong to the same module, say $x, y \in M_1$, but in distinct blocks. Among all such pairs, select x, y so that their distance in the ordering, that is, the size of the set $Z = \{z \mid x \prec z \prec y\}$ is minimized. Let X, Y be the blocks that contain x, y respectively. Note that by the selection of x, y we have that x is the last vertex of X, y is the first vertex of Y, $X \cup Y \subseteq M_1$, and $M_1 \cap Z = \emptyset$.

Let $d^Z_{\text{out}}(x)$ (respectively $d^Z_{\text{in}}(x)$) be the out-degree (respectively in-degree) of x towards the set Z. Because M_1 is a module, all vertices of M_1 have the same in-degree and out-degree towards Z, and in particular, $d^Z_{\text{out}}(x) = d^Z_{\text{out}}(y)$ and $d^Z_{\text{in}}(x) = d^Z_{\text{in}}(y)$. Now, if $d^Z_{\text{in}}(x) > d^Z_{\text{out}}(x)$, we can obtain an ordering that implies a smaller fas by placing x immediately after the last vertex of Z. This would contradict the optimality of $\prec$, so it must be impossible. Similarly, if $d^Z_{\text{in}}(x) < d^Z_{\text{out}}(x)$, we have $d^Z_{\text{in}}(y) < d^Z_{\text{out}}(y)$, and we can obtain a strictly better ordering by placing y immediately before the first vertex of Z, contradiction. We conclude that $d^Z_{\text{in}}(x) = d^Z_{\text{in}}(y)$. Therefore, moving all the vertices of X so that they appear immediately after the last vertex of Z produces an ordering which is equally good as the current one, is still winning for v, and has a smaller number of blocks. ◀

2.1 Complexity

We recall the class Θ_2^p which is known to have several equivalent characterizations, including $\text{P}^{\text{NP}[\log n]}$ (P with the right to make $O(\log n)$ queries to an NP oracle), L^{NP} (logarithmic-space Turing machines with access to an NP oracle), and $\text{P}^{\text{NP}}_{||}$ (P with parallel non-adaptive access to an NP oracle). We refer the reader to [13] for more information on this class. In [17] it was shown that the following problem is Θ_2^p-complete: given a graph G, is the maximum clique size $\omega(G)$ odd? In [10] it is mentioned that the following problem, called Max Model, is Θ_2^p-complete: given a satisfiable CNF formula ϕ containing a special variable x, is there a satisfying assignment of ϕ that sets x to True and has maximum Hamming weight (among all satisfying assignments), where the Hamming weight of an assignment is the number of variables it sets to True.

We will use as a starting point for our reduction a variant of Max Model which we show is Θ_2^p-complete below. The main difference between this variant and the standard version is that we assume that the given formula is satisfied by the assignment that sets all variables to False.

▶ **Lemma 2.** *The following problem is Θ_2^p-complete. Given a 3-CNF formula ϕ containing a distinguished variable x, such that ϕ is satisfied by the all-False assignment, decide if there exists a satisfying assignment for ϕ that sets x to True and has maximum weight among all satisfying assignments.*

Proof. We start with a graph $G = (V, E)$ for which the question is if the maximum independent set has odd size (clearly this is equivalent to the question of deciding if the maximum clique has odd size by taking the complement of G, so our starting problem is Θ_2^p-complete [17]). Let $|V| = n$ and suppose $V = \{v_1, \dots, v_n\}$. We construct a formula ϕ as follows: for each $i \in \{1, \dots, n\}$ we build $(n+1)$ variables $x_i^1, \dots, x_i^{n+1}$ and for each $j, k \in \{1, \dots, n+1\}$ we add the clause $(x_i^j \to x_i^k)$; for each $(v_{i_1}, v_{i_2}) \in E$, for each $j, k \in \{1, \dots, n+1\}$ we add the clause $(\neg x_{i_1}^j \vee \neg x_{i_2}^k)$; we construct n variables $y_1, \dots, y_n$ and clauses that represent the constraints $(y_1 = x_1^1)$, and for each $i \in \{2, \dots, n\}$, $(y_i = y_{i-1} \oplus x_i^1)$. We set y_n as the distinguished variable of ϕ. The formula construction can clearly be carried out in polynomial time, and no clause has size more than three. Furthermore, setting everything to False satisfies all clauses. Intuitively, for each vertex we have constructed $n+1$ variables that will be set to True if we take this vertex in the independent set. The first set of clauses ensures that we make a consistent choice among the copies; the second set that we indeed select an independent set; and the third calculates the parity of its size.

We now observe that independent sets S of G naturally correspond to satisfying assignments of ϕ. In particular, given an independent set $S \subseteq V$ we can construct an assignment by setting, for all i, j, x_i^j to True if and only if $v_i \in S$; we then complete the assignment by giving appropriate values to the y_i variables so that the parity constraints are satisfied. For the converse direction, we can extract an independent set S from a satisfying assignment by setting $v_i \in S$ if and only if the assignment sets x_i^1 to True. We observe the y_n is set to True in a satisfying assignment if and only if the corresponding independent set has odd size (indeed, for each i, y_i is set to True if the intersection of the independent set with the first i vertices has odd size).

Suppose now that there exists an independent set S of maximum size k and that k is odd. Then, there exists a satisfying assignment of maximum weight that sets y_n to True. Indeed, suppose for contradiction that the maximum satisfying assignment σ sets y_n to False. Then, the corresponding independent set S' must have even size k'. Since k is odd and S is a maximum independent set, $k' < k$. But then, the weight of σ is at most $k'(n+1) + n < k(n+1)$. However, the assignment corresponding to S has weight at least $k(n+1)$, contradiction.

For the converse direction, suppose there exists a satisfying assignment σ of maximum weight that sets y_n to True. The corresponding independent set S has odd size, say $|S| = k$. If there exists a maximum independent set S' that has even size k', then $k' > k$. However, the corresponding truth assignment σ' would have weight at least $k'(n+1) > k(n+1) + n$. Since σ has weight at most $k(n+1) + n$ we get a contradiction to the optimality of σ.

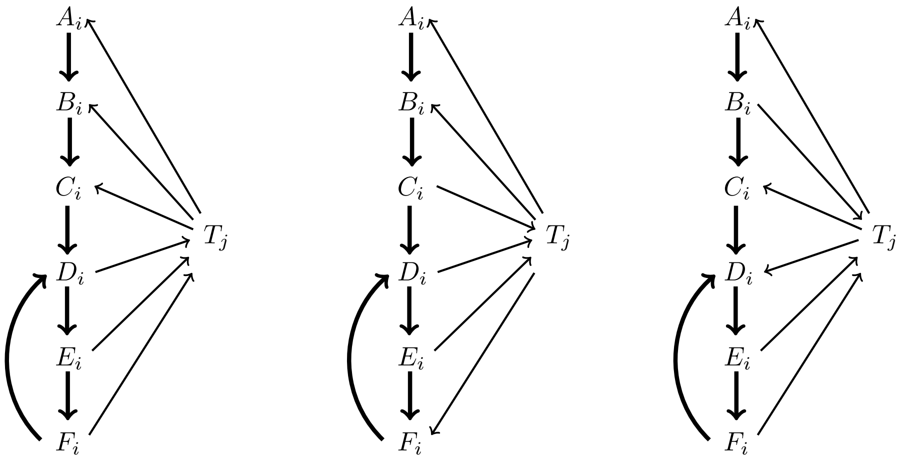
We conclude that there is a satisfying assignment to ϕ of maximum weight that sets y_n to True if and only if the maximum independent set of G has odd size. ◀

3 Reduction to Slater

This section presents the main result of the paper, stated in Theorem 3. The theorem is based on a reduction from the problem of Lemma 2 to the problem of deciding if a vertex of a tournament is a Slater winner. Before we dive into the proof, let us give some high level intuition (we also invite the reader to take a look at Figure 1).

We will build a tournament to represent a CNF formula ϕ with n variables and m clauses by constructing n groups of "large" modules $(A_i, B_i, C_i, D_i, E_i, F_i$, for $i \in \{1, \dots, n\})$ and m "small" modules T_j for $j \in \{1, \dots, m\}$. The internal structure of the modules will be irrelevant and we only care about their ordering, which we may assume to be contiguous thanks to Lemma 1. We will make sure to adjust the sizes of the modules and their connections so that we have the following properties:

1. In any reasonable ordering, all six large modules representing variable x_i come before the six modules representing x_{i+1}. This will naturally order the large modules into n sections.
2. Inside a section, any reasonable ordering will place A_i, B_i, C_i first. Then, if the remaining modules are ordered $D_i \prec E_i \prec F_i$, this encodes that x_i is set to True.



(a) x_i does not appear in c_j. **(b)** $x_i \in c_j$. **(c)** $\neg x_i \in c_j$.

Figure 1 Gadgets of the reduction of Theorem 3. On the left of each figure the six large modules $A_i, B_i, C_i, D_i, E_i, F_i$ represent the variable x_i. Missing (thick) arcs go downwards, so the ordering is forced except for the last three modules. The depicted ordering $D_i \prec E_i \prec F_i$ encodes that x_i is True. In the three figures we depict the connections between the six modules and the small module T_j representing clause c_j depending on whether x_i appears in c_j. In the first case placing T_j anywhere costs at least three arcs. In the second case, placing T_j after E_i costs two arcs, because setting x_i to True satisfies c_j. In the last case, a similarly advantageous placement could be obtained by using the ordering $E_i \prec F_i \prec D_i$ (which encodes that x_i is False) and putting T_j before D_i.

3. Connections between T_j and variable modules will be such that if T_j is placed completely before or completely after the section of a variable x_i, then the cost is the same. However, the cost may be lower if T_j is placed inside the section of x_i. In that case, we must check if x_i appears in the clause c_j in the original formula and the ordering of the section of x_i encodes an assignment to x_i that satisfies c_j.
4. Variable modules are so large that the ordering must always encode a satisfying assignment to the formula (which exists by assumption). The ordering of T_j modules among themselves is irrelevant.
5. In order to encode the weight of a satisfying assignment, we make E_i modules slightly larger (we add 2 extra vertices). Then, the ordering $D_i \prec E_i \prec F_i$, which encodes that x_i is True, is better than other orderings that encode satisfying assignments. Hence, the optimal ordering will represent a satisfying assignment to ϕ with maximum weight.
6. Finally, in order to encode that there is a special variable x_n which must be set to True, we add one extra vertex to E_n. This makes sure that setting x_n to True is more advantageous than setting any other variable to True, but not more advantageous than setting two other variables to True.

Armed with the intuition of the previous list, we are now ready to present all the details of our reduction.

▶ **Theorem 3.** *The following problem is Θ_2^p-complete: given a tournament $T = (V, A)$ and a vertex $v \in V$, decide if v is a Slater winner.*

Proof. We perform a reduction heavily inspired by the reduction of [9] proving that computing the minimum fas of a tournament is NP-complete, though we include a minor modification proposed by Bachmeier et al. [3] which will later allow us to show that our instances are realizable using seven voters. The main complication compared to the reductions of [9, 3] is that we now need to encode the CNF formula in a way that satisfying assignments of larger weight correspond to orderings with better objective value.

We start with a formula ϕ, as given in Lemma 2. Let $x_1, \ldots, x_n$ be the variables of ϕ and suppose that the question is whether there exists a satisfying assignment of maximum weight that sets x_n to True. Recall that by assumption the all-False assignment satisfies ϕ. Let m be the number of clauses of ϕ.

We define two numbers s_1, s_2 which satisfy the following properties:

$$\begin{aligned} s_1^2 &> (3n-1)ms_1s_2 + 3ns_1 + m^2s_2^2 + 9m(n-1)s_2 && (1)\\ s_1s_2 &> 3ns_1 + m^2s_2^2 + 9m(n-1)s_2 && (2)\\ s_1 &> m^2s_2^2 + 9m(n-1)s_2 && (3)\end{aligned}$$

For concreteness, set $s_2 = (n+m)^5$ and $s_1 = s_2^5 = (n+m)^{25}$ and the above inequalities are easily satisfied when $n+m$ is sufficiently large. Importantly, s_1, s_2 are polynomially bounded in $n+m$. Intuitively, the idea is that s_1, s_2 are two very large numbers, and s_1 is significantly larger. We will construct modules of size (roughly) s_1 or s_2, and the values are chosen so that arcs between large modules will be very important, arcs between large and small modules quite important, and arcs between small modules almost irrelevant.

We now construct our tournament as follows: for each $i \in \{1, \ldots, n\}$ we construct 6 modules, call them $A_i, B_i, C_i, D_i, E_i, F_i$. Modules A_i, B_i, C_i, D_i, F_i have size s_1, while modules E_i have size $s_1 + 2$ if $i < n$, and the size of E_n is $s_1 + 3$. Internally, each of these modules induces a transitive tournament. For $i < j$ we add all arcs from $A_i \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$ to $A_j \cup B_j \cup C_j \cup D_j \cup E_j \cup F_j$. For each $i \in \{1, \ldots, n\}$ we add all possible arcs (i) from A_i to $B_i \cup C_i \cup D_i \cup E_i \cup F_i$ (ii) from B_i to $C_i \cup D_i \cup E_i \cup F_i$ (iii) from C_i to $D_i \cup E_i \cup F_i$ (iv) from D_i to E_i (v) from E_i to F_i (vi) from F_i to D_i. The graph we have constructed so far is a tournament with n sections, each made up of 6 modules. Each such section represents a variable x_i and the sections are linearly ordered. The structure inside each section is essentially the transitive closure of $A_i \to B_i \to C_i \to D_i \to E_i \to F_i$ with the exception that arcs between D_i and F_i are heading towards D_i.

We now complete the construction by adding to the current tournament some vertices that represent the clauses of ϕ. In particular, for each $j \in \{1, \ldots, m\}$ we construct a module T_j of size s_2 to represent the j-th clause of ϕ. Internally, T_j is a transitive tournament. For $j, j' \in \{1, \ldots, m\}$, the arcs between T_j and $T_{j'}$ are set in an arbitrary direction. What remains is to explain how the arcs between T_j and the modules representing the variables are set so as to encode the incidence of variables with clauses. For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ we do the following:

1. If x_i does not appear in the j-th clause we add all arcs from T_j to $A_i \cup B_i \cup C_i$ and all arcs from $D_i \cup E_i \cup F_i$ to T_j.
2. If x_i appears positive in the j-th clause we add all arcs from T_j to $A_i \cup B_i \cup F_i$ and all arcs from $C_i \cup D_i \cup E_i$.
3. If x_i appears negative in the j-th clause we add all arcs from T_j to $A_i \cup C_i \cup D_i$ and all arcs from $B_i \cup E_i \cup F_i$.

This completes the construction and the question we want to answer is whether the last vertex (that is, the sink) of the transitive tournament induced by F_n is a Slater winner of the whole graph.

We need to prove that the designated vertex is a Slater winner if and only if there is a satisfying assignment for ϕ with maximum weight that sets x_n to True. We will do this by establishing some properties regarding any optimal ordering of the constructed tournament, showing that such an ordering must always have a structure which implies a satisfying assignment of ϕ with maximum weight. We will rely heavily on Lemma 1, since the tournament we have constructed can be decomposed into $6n+m$ modules, namely, A_i, B_i, C_i, D_i, E_i, and F_i, for $i \in \{1, \dots, n\}$, and T_j, for $j \in \{1, \dots, m\}$. We therefore assume without loss of generality that these sets are placed contiguously in an optimal ordering.

Let us first argue that any optimal ordering must have some desirable structure which necessarily encodes a satisfying assignment for ϕ. To do this it will be helpful to start with a baseline ordering and calculate its implied fas, as then any ordering which implies a larger fas will be necessarily suboptimal. Consider the ordering which is defined as $A_i \prec B_i \prec C_i \prec E_i \prec F_i \prec D_i$ for each $i \in \{1, \dots, n\}$ and which sets $D_i \prec A_{i+1}$, where each module is internally ordered in the optimal way. We insert into this ordering of the modules that represent variables, the modules T_j as follows: for each $j \in \{1, \dots, m\}$, we find a variable x_i that appears negative in the j-th clause (such a variable must exist, since ϕ is satisfied by the all-False assignment), and place all of T_j between F_i and D_i. If for some pair $T_j, T_{j'}$ their relative ordering is not yet fully specified, we order them in some arbitrary way.

The arcs incompatible with the above ordering are (i) the at most $ns_1(s_1+3)$ arcs going from a module D_i to a module E_i (ii) for each T_j that was placed between F_i and D_i we have $2s_1s_2$ arcs (towards $A_i \cup C_i$), as well as at most $3(s_1+3)s_2$ arcs to each other group $A_{i'} \cup B_{i'} \cup C_{i'} \cup D_{i'} \cup E_{i'} \cup F_{i'}$, for $i' \neq i$ (iii) the total number of arcs between modules T_j is at most $m^2 s_2^2$. Therefore, we have that the fas implied by this ordering has size at most

$$\begin{aligned} B &\leq ns_1(s_1+3) + m(2s_1s_2 + 3(n-1)(s_1+3)s_2) + m^2s_2^2 = \\ &= ns_1^2 + (3n-1)ms_1s_2 + 3ns_1 + m^2s_2^2 + 9m(n-1)s_2 \end{aligned}$$

In the remainder we will therefore only consider orderings which imply a fas of size at most B, as other orderings are suboptimal. This allows us to draw some conclusions regarding the structure of an optimal ordering. First, observe that for each $i \in \{1, \dots, n\}$, any ordering of $D_i \cup E_i \cup F_i$ will contribute at least s_1^2 arcs to the fas. Using inequality (1), we have that there are at most n pairs of "large" modules (that is, modules of size at least s_1) which are incorrectly ordered, that is, ordered so that all arcs between the modules are included in the fas. Indeed, if there are $n+1$ such pairs, the fas will have size at least $(n+1)s_1^2 > B$. We conclude that regarding the $6n$ large modules we must have the following ordering:

1. For each $i < j$, we have that all vertices of $A_i \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$ (the section that represents the variable x_i) are before all vertices of $A_j \cup B_j \cup C_j \cup D_j \cup E_j \cup F_j$ (the section that represents the variable x_j).
2. For each $i \in \{1, \dots, n\}$, we have $A_i \prec B_i \prec C_i$ and all vertices of $A_i \cup B_i \cup C_i$ are before $D_i \cup E_i \cup F_i$.
3. For each $i \in \{1, \dots, n\}$ we have $D_i \prec E_i \prec F_i$, or $E_i \prec F_i \prec D_i$, or $F_i \prec D_i \prec E_i$.

We would now like to construct a correspondence between assignments to ϕ and orderings of the tournament that respect the above conditions. On the one hand, if we are given an assignment σ we construct an ordering of the variable sections as above and for each i, if σ set x_i to True we set $D_i \prec E_i \prec F_i$, otherwise we set $E_i \prec F_i \prec D_i$. In the converse direction, given an ordering that respects the above conditions (which any optimal ordering must do), we extract an assignment by setting, for each i, x_i to True if and only if $D_i \prec E_i \prec F_i$.

We now argue that the assignment corresponding to an optimal ordering must also be satisfying for ϕ, as otherwise the fas will have size strictly larger than B, contradicting the optimality of the ordering. For the sake of contradiction, suppose we have an optimal ordering which corresponds to an assignment falsifying a clause. As argued above, there are at least ns_1^2 arcs in the fas contributed by the ordering of the large modules, so we concentrate on the modules T_j representing clauses. A module T_j representing any clause must be incident on at least $(3n-1)s_1s_2$ arcs of the fas connecting it to large modules. To see this, consider the following: we will say that T_j is in the interior of section i, if $A_i \prec T_j$ and T_j is placed before one of D_i, E_i, or F_i. T_j can be in the interior of at most one section i, so for each $i' \neq i$ we observe that at least $3s_1s_2$ arcs incident on T_j and modules of the group i' are in the fas. This gives $3(n-1)s_1s_2$ arcs. In addition, no matter where we place T_j in the interior of section i, at least a further $2s_1s_2$ arcs of the fas are obtained: if T_j is after C_i, then we get the arcs to $A_i \cup B_i$ or the arcs to $A_i \cup C_i$; if T_j is between A_i and C_i, we get the arcs to A_i and at least $2s_1s_2$ arcs from $D_i \cup E_i \cup F_i$. Hence, we get at least $(3n-1)s_1s_2$ arcs in the fas for each T_j.

Furthermore, suppose that the assignment corresponding to the ordering does not satisfy the j-th clause. Then, we claim that at least $3ns_1s_2$ arcs connecting T_j to large modules are included in the fas. Indeed, if T_j is in the interior of section i, it can either be before or after C_i. If it is before C_i, as we observed in the previous paragraph, we always have at least $3s_1s_2$ arcs in the fas between T_j and the large modules of section i. If T_j is placed after C_i, we have the following cases: (i) if x_i does not appear in the clause, then at least $3s_1s_2$ arcs between T_j and the large modules of section i are in the fas (ii) if x_i appears positive in the j-th clause, we know that F_i is not placed last in section i (otherwise the assignment would satisfy the j-th clause), so wherever we place T_j, at least $3s_1s_2$ arcs are included in the fas (iii) similarly if x_i appears negative, since the assignment does not satisfy the clause, F_i is last, so again at least $3s_1s_2$ arcs are included in the fas.

From the above calculations, if the assignment that corresponds to an ordering falsifies a clause, the fas has size at least $ns_1^2+(m-1)(3n-1)s_1s_2+3ns_1s_2 = ns_1^2+(3n-1)ms_1s_2+s_1s_2 > B$, where we used inequality (2). We conclude that an optimal ordering must correspond to a satisfying assignment.

We now need to argue that the assignment corresponding to an optimal ordering of the tournament must be a satisfying assignment of maximum weight. Suppose for contradiction that the assignment corresponding to an optimal ordering, call it σ_1, sets k variables to True, but there exists another satisfying assignment, call it σ_2, that sets at least $k+1$ variables to True. We will show that starting from σ_2 we can obtain a better ordering of the tournament, contradicting the optimality of the original ordering.

We claim that the ordering from which we extracted σ_1 includes at least $ns_1^2 + (3n-1)ms_1s_2 + 2(n-k)s_1$ arcs in the fas. This is because in the section corresponding to the $n-k$ variables that σ_1 sets to False, either the arcs from E_i to F_i, or the arcs from D_i to E_i are in the fas (since F_i is not placed last in the section), and these are at least $s_1(s_1+2) = s_1^2 + 2s_1$ arcs.

We construct an ordering from σ_2 as follows: we order the variable section in the normal way and inside each section, if $\sigma_2(x_i) = \text{True}$ we use the ordering $D_i \prec E_i \prec F_i$, otherwise we use the ordering $E_i \prec F_i \prec D_i$. For each T_j, we find a variable x_i that satisfies the j-th clause and place T_j in section i immediately before the last module of this section. If for j, j' the order of $T_j, T_{j'}$ is not implied by the above, we set it arbitrarily. The fas implied by this ordering has size at most

$$\begin{aligned} B' &\leq ns_1^2 + 2(n-k-1)s_1 + s_1 + m(2s_1s_2 + 3(n-1)(s_1+3)s_2) + m^2s_2^2 = \\ &= ns_1^2 + 2(n-k)s_1 + (3n-1)ms_1s_2 - s_1 + m^2s_2^2 + 9m(n-1)s_2 \end{aligned}$$

Here, the calculations for the terms $m(2s_1s_2 + 3(n-1)(s_1+3)s_2) + m^2s_2^2$ are the same as in the calculation of B; the term $2(n-k-1)s_1$ takes into account that there are $n-k-1$ sections that correspond to variables set to False; and the s_1 term is due to the fact that

x_n may be one of the variables set to False and E_n has size $s_1 + 3$ and not $s_1 + 2$. Using inequality (3) we have that $-s_1+m^2s_2^2+9m(n-1)s_2 < 0$, so the ordering we have constructed from σ_2 is better than the one from which we extracted σ_1, contradiction.

At this point we are almost done because we have argued that an optimal ordering of the tournament corresponds to a satisfying assignment of maximum weight and furthermore, since the correspondence sets x_n to True if and only if F_n is the last module in the ordering, the sink of F_n will be last in the ordering if and only if the assignment sets x_n to True. However, ϕ could have several satisfying assignments of the same weight, and since we have set arcs between T_j modules arbitrarily, it could be the case that a maximum weight assignment that sets x_n to False results in a better ordering, making another vertex the Slater winner. This is the reason why we have set E_n to be slightly larger than all other modules E_i, so that setting x_n to True is always slightly more advantageous than setting any other variable to True.

Concretely, we argue the following: any optimal ordering of the tournament corresponds to a satisfying assignment of ϕ with maximum weight; and furthermore if a satisfying assignment of ϕ with maximum weight sets x_n to True, then any optimal ordering places F_n last. We need to argue the second claim, so suppose for contradiction that an optimal ordering does not place F_n last and that the assignment that corresponds to this ordering is σ_1. Furthermore, suppose that there exists a satisfying assignment σ_2 of maximum weight that sets x_n to True. Say that both σ_1, σ_2 set k variables to True.

We first observe that the ordering from which we extracted σ_1 implies a fas of size at least $ns_1^2 + 2(n-k)s_1 + s_1 + (3n-1)ms_1s_2$. This is because there are $(n-k-1)$ sections where the fas contains $s_1(s_1+2)$ arcs incident on a module E_i, k sections where the fas contains s_1^2 arcs incident from F_i to D_i, and in the section corresponding to x_n the fas contains $s_1(s_1+3)$ arcs, incident on E_n.

On the other hand, if we construct an ordering from σ_2 in the same way as we did previously, the fas obtained will have size at most

$$\begin{aligned} B'' &\leq ns_1^2 + 2(n-k)s_1 + m(2s_1s_2 + 3(n-1)(s_1+3)s_2) + m^2s_2^2 = \\ &= ns_1^2 + 2(n-k)s_1 + (3n-1)ms_1s_2 + m^2s_2^2 + 9m(n-1)s_2 \end{aligned}$$

Again, using inequality (3) which states that $s_1 > m^2s_2^2 + 9m(n-1)s_2$ we conclude that the new ordering is better, contradicting the optimality of the original ordering.

We now summarize our arguments: we have shown that any optimal ordering of the tournament always corresponds to a maximum weight satisfying assignmet of ϕ and furthermore, it corresponds to a maximum weight satisfying assignment that sets x_n to True if this is possible; furthermore, if an optimal ordering corresponds to an assignment that sets x_n to True then the last vertex of F_n is a Slater winner. We therefore have two cases: if the last vertex of F_n is a Slater winner, then since optimal orderings give rise to satisfying assignments of maximum weight, there is a maximum weight satisfying assignment of ϕ setting x_n to True; if the last vertex of F_n is not a Slater winner, then the maximum weight satisfying assignment we extract from an optimal ordering sets x_n to False, and there is no satisfying assignment of the same weight setting x_n to True. We conclude that determining if a vertex is a Slater winner is equivalent to deciding if ϕ has a maximum weight satisfying assignment setting x_n to True, and is therefore Θ_2^p-complete. ◀

4 Hardness for 7 Voters

In this section we show that the tournaments constructed in Theorem 3 correspond to instances that could result from the aggregation of the preferences of 7 voters and as a result the problem of determining a Slater winner remains Θ_2^p-complete even for 7 voters. Our approach follows along the lines of the arguments of Bachmeier et al. [3] who proved that determining the Slater winner is NP-hard for 7 voters. Indeed, the proof of [3] consists of an analysis (and tweak) of the construction of Conitzer [9] which establishes that the instances of the reduction can be built by aggregating 7 voter profiles. Since our reduction is very similar to Conitzer's, we essentially only need to adjust the arguments of Bachmeier et al. to obtain Θ_2^p-completeness.

Our first step is to slightly restrict the Θ_2^p-complete problem that is the starting point of our reduction. We present the following strengthening of Lemma 2, which is similar to the problem used as a starting point in the reduction of [3].

▶ **Lemma 4.** *The problem given in Lemma 2 remains Θ_2^p-complete under the following additional restrictions: (i) we are given a partition of the clauses of ϕ in two sets L, R and each variable appears in at most one clause of L and in at most two clauses of R (ii) each literal appears at most once in a clause of R.*

Proof. Given a formula ϕ as in Lemma 2 we construct a new formula ϕ' as follows. Let $x_1, x_2, \ldots, x_n$ be the variables of ϕ and m be the number of its clauses. For each x_i, $i \in \{1, \ldots, n\}$ we construct m variables, call them $y_i^1, y_i^2, \ldots, y_i^m$. For each $i \in \{1, \ldots, n\}$, for $j \in \{1, \ldots, m-1\}$ we construct the clause $(\neg y_i^j \vee y_i^{j+1})$, as well as the clause $(\neg y_i^m \vee y_i^1)$. Let R be the set of clauses constructed so far and note that each literal appears at most once and each variable at most twice in these clauses. Intuitively, the clauses of R ensure that for each i, all variables in the set $\{y_i^1, y_i^2, \ldots, y_i^m\}$ must receive the same value in a satisfying assignment.

Now, we consider the clauses of ϕ one by one. If the j-th clause contains the variable x_i, we replace it by the variable y_i^j. Doing this for all clauses of ϕ we obtain a set of clauses, call it L, where each variable appears at most once (assuming without loss of generality that clauses of ϕ have no repeated literals).

If x_n was the designated variable of ϕ we set y_n^1 as the designated variable of ϕ'. It is now not hard to make a correspondence between satisfying assignments of ϕ and ϕ' (x_i is set to True if all y_i^j are set to True) in a way that preserves weights (the weight of an assignment to ϕ' is m times the weight of the corresponding assignment for ϕ). Hence, determining if a maximum weight satisfying assignment to ϕ' sets y_n^1 to True is Θ_2^p-complete. Observe also that ϕ' is satisfied by the all-False assignment. ◀

We now obtain the result of this section by starting the reduction of Theorem 3 from the problem of Lemma 4. In the statement of the theorem below, when we say that a tournament $T = (V, A)$ can be obtained from 7 voters, we mean that there exist 7 total orderings of V such that for all $(a, b) \in A$ we have that $a \prec b$ in at least 4 of the orderings.

▶ **Theorem 5.** *Determining if a vertex of a tournament is a Slater winner remains Θ_2^p-complete even for tournaments that can be obtained from 7 voters.*

Proof. We perform the same reduction as in Theorem 3 except we start from the special case given in Lemma 4. What remains is to show that the instance we construct can result from aggregating 7 orderings. Recall that our tournament contains $6n$ modules representing the variables, called $A_i, B_i, C_i, D_i, E_i, F_i$, for $i \in \{1, \ldots, n\}$ and m modules representing the clauses, called T_j, for $j \in \{1, \ldots, m\}$. Since modules are internally transitive, we will assume that the 7 voters have preferences which agree with the directions of the arcs inside the modules and hence we focus on the arcs between modules. Recall that in the reduction of Theorem 3, arcs between modules T_j are set arbitrarily. To ease presentation, assume that when $j < j'$ we have the arcs $T_j \to T_{j'}$.

The first voter has preferences $A_1 \prec B_1 \prec C_1 \prec D_1 \prec E_1 \prec F_1 \prec A_2 \ldots \prec F_n \prec T_1 \prec T_2 \prec \ldots \prec T_m$. In other words, the first voter orders all the variable modules before all the clause modules, orders variable groups according to their index, and inside each variable group she has the ordering $A_i \prec B_i \prec C_i \prec D_i \prec E_i \prec F_i$.

We now add two voters with the intent of constucting all the arcs of the set

$$X_0 = \left(\bigcup_j T_j \times \bigcup_i (A_i \cup B_i \cup C_i) \right) \cup \bigcup_i (F_i \times D_i)$$

The first of these voters has ordering $(E_1 \prec E_2 \prec \ldots \prec E_n) \prec (F_1 \prec D_1 \prec F_2 \prec D_2 \prec \ldots \prec F_n \prec D_n) \prec (T_1 \prec T_2 \prec \ldots \prec T_m) \prec (A_1 \prec B_1 \prec C_1 \prec A_2 \prec B_2 \prec C_2 \prec \ldots \prec A_n \prec B_n \prec C_n)$. The second of these voters has ordering $(T_m \prec T_{m-1} \prec \ldots \prec T_1) \prec (C_n \prec B_n \prec A_n \prec C_{n-1} \prec B_{n-1} \prec A_{n-1} \prec \ldots \prec C_1 \prec B_1 \prec A_1) \prec (F_n \prec D_n \prec F_{n-1} \prec D_{n-1} \prec \ldots \prec F_1 \prec D_1) \prec (E_n \prec E_{n-1} \prec \ldots \prec E_1)$. Note that these two voters agree that all modules of $\bigcup_j T_j$ come before all modules of $\cup_i (A_i \cup B_i \cup C_i)$ and that for each i we have $F_i \prec D_i$, but disagree on every other pair of modules, hence the two voters together induce exactly the set of arcs X_0 cited above.

If we now consider the three voters we have so far, we observe that much of our construction is already induced:

1. For $i < i'$ we have arcs from $A_i \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$ to $A_{i'} \cup B_{i'} \cup C_{i'} \cup D_{i'} \cup E_{i'} \cup F_{i'}$ because of the preferences of the first voter, as the other two voters disagree on these arcs.
2. For each $i \in \{1, \ldots, n\}$, inside the group $A_i \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$, we have arcs that agree with the ordering $A_i \prec B_i \prec C_i \prec D_i \prec E_i \prec F_i$, except that we have arcs from F_i to D_i. This is because the second and third voter agree that $F_i \prec D_i$, but disagree on every other pair (hence the preferences of the first voter prevail for the other pairs).
3. For each $j < j'$ we have arcs from T_j to $T_{j'}$, due to the preferences of the first voter, as the other two disagree.
4. For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ we have arcs from T_j to $A_i \cup B_i \cup C_i$, because the second and third voter agree that $T_j \prec (A_i \cup B_i \cup C_i)$ (though the first voter disagrees).
5. For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ we have arcs from $D_i \cup E_i \cup F_i$ to T_j, because the second and third voter disagree on these pairs, so the preferences of the first voter break the tie.

We therefore have that the tournament that follows from aggregating the preferences of the first three voters almost corresponds to the one we want to construct, except that for each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ the arcs between T_j and $A_i \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$ correspond to the arcs we would want if x_i did not appear in the j-th clause (in other words, the three voters we have so far induce the general structure of the construction, but do not encode which variable appears in which clause). Furthermore, if we look at the relationship between any two modules so far, the margin of victory is always exactly one (that is, there do not exist two modules X, Y such that all three voters agree that $X \prec Y$).

Hence, what remains is to use the four remaining voters to "fix" this, so that if x_i appears (positive or negative) in the j-th clause, we have the arcs prescribed in the reduction of Theorem 3. We will achieve this by giving two pairs of voters. Each pair of voters will disagree on all pairs of modules except a specific set of arcs that we want to fix. Hence, adding the pair of voters to the electorate will repair the arcs in question (since the current margin of victory for all arcs is one), while leaving everything else unchanged.

Recall that the clause set is given to us partitioned into two sets R, L so that each variable appears in at most one clause of L and each literal in at most one clause of R. We will use the slightly weaker property that each literal appears at most once in each of L, R. Abusing notation we will write $j \in R$ if the j-th clause is in R (similarly for $j \in L$). We will also write $x_i \in c_j$ (respectively $\neg x_i \in c_j$) if x_i appears positive (respectively negative) in the j-th clause.

Consider now the following two sets of arcs:

$$X_1 = \left(\bigcup_{j \in R} \cup_{i:x_i \in c_j} (T_j \times F_i) \cup \cup_{i:\neg x_i \in c_j} (T_j \times D_i) \right) \cup \left(\bigcup_{j \in L} \cup_{i:x_i \in c_j} (C_i \times T_j) \cup \cup_{i:\neg x_i \in c_j} (B_i \times T_j) \right)$$

$$X_2 = \left(\bigcup_{j \in L} \cup_{i:x_i \in c_j} (T_j \times F_i) \cup \cup_{i:\neg x_i \in c_j} (T_j \times D_i) \right) \cup \left(\bigcup_{j \in R} \cup_{i:x_i \in c_j} (C_i \times T_j) \cup \cup_{i:\neg x_i \in c_j} (B_i \times T_j) \right)$$

Our plan is to give a pair of voters whose preferences induce the arcs of X_1 and another pair whose preferences induce the arcs of X_2. Here when we say that two voters induce a set of arcs X we mean that for each $(a, b) \in X$ both voters have $a \prec b$ and for each $(a, b) \notin X$ one voter has $a \prec b$ and the other has $b \prec a$. Before we proceed we observe that if X_1, X_2 are inducible by a pair of voters each, then adding these four voters to the three voters we have described so far produces the tournament of Theorem 3. Indeed, suppose that x_i appears positive in clause c_j and $j \in R$. Then, if we consider the arcs in the tournament induced by the first three voters, we need to inverse the arcs between T_j and F_i (which currently point $F_i \to T_j$), and the arcs between T_j and C_i (which currently point $T_j \to C_i$). But the arcs between T_j and F_i are inversed thanks to X_1, while the arcs between T_j and C_i are inversed thanks to X_2, where we use the fact that X_1, X_2 represent the consensus of two voters, while the margin of victory for any arc induced by the first three voters is one. Similar arguments apply if x_i appears negative in c_j, or $j \in L$. Hence, if a pair of voters induces X_1 and another induces X_2, taking the union of these four voters with the three voters we have described produces the tournament of Theorem 3 and completes the proof.

We now recall that it was shown in [3] that X_1, X_2 are inducible by two voters each, since these sets of arcs are unions of stars (if we contract each module to a vertex). Let us explain in more detail how to represent X_1 as the union of the preferences of two voters (the arguments for X_2 are essentially identical). We will make use of the fact that each literal appears at most once in L and at most once in R.

We will say that a module from a variable group is "active" if it is incident on an arc of X_1. In particular, modules A_i, E_i, for $i \in \{1, \dots, n\}$ are not active, and neither are modules F_i such that x_i does not appear positive in R (and similarly for B_i, C_i, D_i). We will concentrate on the ordering of active modules because if we find two voter profiles that order these modules in a way that induces X_1, we can add an arbitrary ordering of the inactive modules in the beginning of the preferences of the first voter, and the opposite of that ordering at the end of the preferences of the second voter. This will have as effect that the two voters disagree on any pair that involves an element of an inactive module, as desired.

We now observe that for each active module M from $\bigcup_i B_i \cup C_i \cup D_i \cup F_i$, there exists exactly one T_j such that M has arcs to T_j in X_1. This is because every literal appears in at most one clause of R and at most one clause of L. Now, we construct two voter profiles as follows: one voter orders the T_j modules in increasing order of index and the other in decreasing order. For each active module D_i or F_i, we insert the module immediately after the T_j from which the module receives arcs in X_1 in both orderings; for active modules B_i or C_i we insert them immediately before the T_j towards which the module has arcs in both orderings. Note that this does not fully specify the ordering, as if two variables $x_i, x_{i'}$ appear in c_j and $j \in R$, then we need to place F_i and $F_{i'}$ immediately after T_j. We resolve such conflicts by using an arbitrary ordering of the active modules for the first voter and the opposite of that ordering for the second voter, that is, all modules which are supposed to appear immediately after T_j are sorted in one way for the first voter and in the opposite way for the second voter. We now observe that with this ordering for every active module the two voters agree about the arcs connecting the module to its neighboring T_j, while we obtain no other arcs between the module and any other $T_{j'}$ or any other active module. We therefore have two voters whose preferences induce X_1. ◀

References

1 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008.

2 Noga Alon. Ranking tournaments. *SIAM J. Discret. Math.*, 20(1):137–142, 2006.

3 Georg Bachmeier, Felix Brandt, Christian Geist, Paul Harrenstein, Keyvan Kardel, Dominik Peters, and Hans Georg Seedig. k-majority digraphs and the hardness of voting with a constant number of voters. *J. Comput. Syst. Sci.*, 105:130–157, 2019.

4 Felix Brandt, Markus Brill, and Paul Harrenstein. Tournament solutions. In *Handbook of Computational Social Choice*, pages 57–84. Cambridge University Press, 2016.

5 Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia. Introduction to computational social choice. In *Handbook of Computational Social Choice*, pages 1–20. Cambridge University Press, 2016.

6 Richard Chang and Jim Kadin. The Boolean hierarchy and the polynomial hierarchy: A closer connection. *SIAM J. Comput.*, 25(2):340–354, 1996. doi:10.1137/S0097539790178069.

7 Pierre Charbit, Stéphan Thomassé, and Anders Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Comb. Probab. Comput.*, 16(1):1–4, 2007.

8 Irène Charon and Olivier Hudry. An updated survey on the linear ordering problem for weighted or unweighted tournaments. *Ann. Oper. Res.*, 175(1):107–158, 2010.

9 Vincent Conitzer. Computing Slater rankings using similarities among candidates. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 613–619. AAAI Press, 2006. URL: http://www.aaai.org/Library/AAAI/2006/aaai06-098.php.

10 Ronald de Haan. *Parameterized Complexity in the Polynomial Hierarchy – Extending Parameterized Complexity Theory to Higher Levels of the Hierarchy*, volume 11880 of *Lecture Notes in Computer Science*. Springer, 2019. doi:10.1007/978-3-662-60670-4.

11 Ulle Endriss and Ronald de Haan. Complexity of the winner determination problem in judgment aggregation: Kemeny, Slater, Tideman, Young. In *AAMAS*, pages 117–125. ACM, 2015.

12 Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *J. ACM*, 44(6):806–825, 1997.

13 Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Raising NP lower bounds to parallel NP lower bounds. *SIGACT News*, 28(2):2–13, 1997. doi:10.1145/261342.261344.

14 Edith Hemaspaandra, Holger Spakowski, and Jörg Vogel. The complexity of Kemeny elections. *Theor. Comput. Sci.*, 349(3):382–391, 2005.

15 Olivier Hudry. On the complexity of Slater's problems. *Eur. J. Oper. Res.*, 203(1):216–221, 2010.

16 Jörg Rothe, Holger Spakowski, and Jörg Vogel. Exact complexity of the winner problem for Young elections. *Theory Comput. Syst.*, 36(4):375–386, 2003.

17 Klaus W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.*, 51:53–80, 1987. doi:10.1016/0304-3975(87)90049-1.

Improved Ackermannian Lower Bound for the Petri Nets Reachability Problem

Sławomir Lasota
University of Warsaw, Poland

Abstract

Petri nets, equivalently presentable as vector addition systems with states, are an established model of concurrency with widespread applications. The reachability problem, where we ask whether from a given initial configuration there exists a sequence of valid execution steps reaching a given final configuration, is the central algorithmic problem for this model. The complexity of the problem has remained, until recently, one of the hardest open questions in verification of concurrent systems. A first upper bound has been provided only in 2015 by Leroux and Schmitz, then refined by the same authors to non-primitive recursive Ackermannian upper bound in 2019. The exponential space lower bound, shown by Lipton already in 1976, remained the only known for over 40 years until a breakthrough non-elementary lower bound by Czerwiński, Lasota, Lazic, Leroux and Mazowiecki in 2019. Finally, a matching Ackermannian lower bound announced this year by Czerwiński and Orlikowski, and independently by Leroux, established the complexity of the problem.

Our primary contribution is an improvement of the former construction, making it conceptually simpler and more direct. On the way we improve the lower bound for vector addition systems with states in fixed dimension (or, equivalently, Petri nets with fixed number of places): while Czerwiński and Orlikowski prove $\mathcal{F}_k$-hardness (hardness for kth level in Grzegorczyk Hierarchy) in dimension $6k$, our simplified construction yields $\mathcal{F}_k$-hardness already in dimension $3k + 2$.

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Verification by model checking; Theory of computation → Logic and verification

Keywords and phrases Petri nets, reachability problem, vector addition systems

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.46

Related Version Preliminary arXiv report:
Full Version: `https://arxiv.org/abs/2105.08551`

Funding *Sławomir Lasota*: Supported by the ERC project 'Lipa' within the EU Horizon 2020 research and innovation programme (No. 683080).

1 Introduction

Petri nets [41] are an established model of concurrency with extensive and diverse applications in various fields, including modelling and analysis of hardware [6, 27], software [18, 5, 22] and database [4] systems, as well as chemical [1], biological [2] and business [45, 35] processes (the references on applications are illustrative). The model admits various alternative but essentially equivalent presentations, most notably *vector addition systems* (VAS) [24], and *vector addition systems with states* (VASS) [19, Sec.5], [21]. The central algorithmic question for this model is the *reachability problem* that asks whether from a given initial configuration there exists a sequence of valid execution steps reaching a given final configuration. Each of the alternative presentations admits its own formulation of the reachability problem, all of them being equivalent due to straightforward polynomial-time translations that preserve reachability, see e.g. Schmitz's survey [44, Section 2.1]. For instance, in terms of VAS, the problem is stated as follows: given a finite set T of integer vectors in d-dimensional space and two d-dimensional vectors $\mathbf{v}$ and $\mathbf{w}$ of nonnegative integers, does there exist a walk from $\mathbf{v}$ to $\mathbf{w}$ such that it stays within the nonnegative orthant, and every step modifies the current position by adding some vector from T? The model of VASS is a natural extension of VAS with finite control, where $\mathbf{v}$ is additionally equipped with an initial control state, $\mathbf{w}$ with a final one, and each vector in T is additionally equipped with a source-target pair of control states.

We recall, following [44, 9, 10, 11], that importance of the Petri nets reachability problem is widespread, as many diverse problems from formal languages [8], logic [23, 14, 13, 7], concurrent systems [17, 16], process calculi [40], linear algebra [20] and other areas (the references are again illustrative) are known to admit reductions from the VASS reachability problem; for more such problems and a wider discussion, we refer to [44].

Brief history of the problem. The complexity of the Petri nets reachability problem has remained unsettled over the past half century. Concerning the decidability status, after an incomplete proof by Sacerdote and Tenney in 1970s [42], decidability of the problem was established by Mayr [38, 39] in 1981, whose proof was then simplified by Kosaraju [25], and then further refined by Lambert in the 1990s [26]. A different approach, based on inductive invariants, has emerged from a series of papers by Leroux a decade ago [28, 29, 30].

Concerning upper complexity bounds, the first such bound has been shown only in 2015 by Leroux and Schmitz [33], consequently improved to the Ackermannian upper bound [34].

Concerning lower complexity bounds, Lipton's landmark exponential space lower bound from 70ies [36] has remained the state of the art for over 40 years until a breakthrough non-elementary lower bound by Czerwiński, Lasota, Lazic, Leroux and Mazowiecki in 2019 [9] (see also [10]): hardness of the reachability problem for the class Tower of all decision problems that are solvable in time or space bounded by a tower of exponentials whose height is an elementary function of input size. A further refinement of Tower-hardness, in terms of fine-grained complexity classes closed under polynomial-time reductions, has been reported by Czerwiński, Lasota and Orlikowski [11]. Finally, a matching Ackermannian lower bound has been announced recently, independently by Czerwiński and Orlikowski [12], and by Leroux [31] (the two constructions underlying the proofs seem to be significantly different). These results finally close the long standing complexity gap, and yield Ackermann-completness of the Petri nets reachability problem. The techniques used in [12] and [31] substantially differ.

Our contribution. We provide an improvement of the construction of [12]. As our main contribution, we make the construction conceptually simpler and more direct (the idea of improvement is discussed at the end of Section 2, and the central ingredient of our construction is presented in Section 4). Moreover, on the way we improve the parametric lower bound with respect to the dimension of vector addition systems with states (or, equivalently, the number of places of Petri nets[1]). For formulating the result we refer to the complexity classes $\mathcal{F}_\alpha$ corresponding to the Grzegorczyk hierarchy of fast-growing functions [37, 43], indexed by ordinals $\alpha = 0, 1, 2, \ldots, \omega$; for instance, the class $\mathcal{F}_3$ is Tower (class of all decision problems that are solvable in time or space bounded by a tower of exponentials, closed under elementary reductions) and $\mathcal{F}_\omega$ is Ackermann (class of all decision problems that are solvable in time or space bounded by the Ackermann function, closed under primitive-recursive reductions). Results of [12, 31] can be stated in parametric terms as follows: the former shows $\mathcal{F}_k$-hardness of the reachability problem for VASS in dimension $6k$, while the latter one shows $\mathcal{F}_k$-hardness

[1] We remark that a Petri net corresponding to a VASS of dimension d has $d+3$ places, due to 3 extra places for encoding the control states of VASS [21]. Likewise, a VAS corresponding to a VASS of dimension d has dimension $d+3$.

for VASS in dimension $4k+5$. Our simplified construction yields a better lower bound: $\mathcal{F}_k$-hardness already in dimension $3k+2$. This improvement is a step towards establishing the tight dimension-parametric complexity of the problem, as the best known upper bound is $\mathcal{F}_k$-membership in dimension $k-4$ [34]. As a next step, an improvement of the construction of [31] to dimension $2k+4$ has been recently reported in [32].

2 The reachability problem

In this section we define the reachability problem and explain our contribution. Following [9, 10, 11, 12, 31], we work with a convenient presentation of VASS as counter programs without zero tests, where the dimension of a VASS corresponds to the number of counters of a program.

Counter programs. A *counter program* (or simply a *program*) is a sequence of (line-numbered) commands, each of which is of one of the following types:

$\mathsf{x} \mathrel{+}= 1$	(increment counter x)
$\mathsf{x} \mathrel{-}= 1$	(decrement counter x)
goto L **or** L'	(nondeterministically jump to either line L or line L')
zero? x	(zero test: continue if counter x equals 0)

Counters are only allowed to have nonnegative values. We are particularly interested in counter programs *without zero tests*, i.e., ones that use no zero test command. Whenever we use zero tests in the sequel, it is always in view of faithfully simulating them by programs without zero tests.

> Convention: In the sequel, unless specified explicitly, counter programs are implicitly assumed to be without zero tests.

▶ **Example 1.** We write $\mathsf{x} \mathrel{+}= m$ (resp. $\mathsf{x} \mathrel{-}= m$) as a shorthand for for m consecutive increments (resp. decrements) of x. As an illustration, consider the program with three counters $\mathsf{C} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$ (on the left), and its more readable presentation using a syntactic sugar **loop** (on the right):

```
1: goto 2 or 6
2: x -= 1
3: y += 1
4: z += 2
5: goto 1 or 1
6: z += 1
```

```
1: loop
2:     x -= 1
3:     y += 1
4:     z += 2
5: z += 1
```

The program repeats the block of commands in lines 2–4 some number of times chosen nondeterministically (possibly zero, although not infinite because x is decreasing, and hence its initial value bounds the number of iterations) and then increments z. In the sequel we conveniently use **loop** construct instead of explicit **goto** commands. (A dummy command is implicitly added after a **loop** in case it appears at the very end of a program.)

We emphasise that counters are only permitted to have nonnegative values. In the program above, that is why the decrement in line 2 works also as a non-zero test.

Consider a program with counters C. By $\mathbb{N}^{\mathsf{C}}$ we denote the set of all valuations of counters. Given an initial valuation of counters, a *run* (or *execution*) of a counter program is a finite sequence of executions of commands, as expected. A run which has successfully finished we call *complete*; otherwise, the run is *partial.* Observe that, due to a decrement that would cause a counter to become negative, a partial run may fail to continue because it is blocked from further execution. Moreover, due to nondeterminism of **goto**, a program may have various runs from the same initial valuation.

Two programs $\mathcal{P}, \mathcal{Q}$ may be *composed* by concatenating them, written $\mathcal{P}\ \mathcal{Q}$. We silently assume the appropriate re-numbering of lines referred to by **goto** command in $\mathcal{Q}$.

The reachability problem. Given a subset $R \subseteq \mathbb{N}^{\mathsf{C}}$ of valuations, by a run *from* R we mean any run whose initial valuation belongs to R. A complete run is called X-*zeroing*, for a subset $\mathsf{X} \subseteq \mathsf{C}$ of counters, if it ends with $\mathsf{x} = 0$ for all $\mathsf{x} \in \mathsf{X}$. When $\mathsf{X} = \{\mathsf{x}\}$ and/or $R = \{r\}$ are a singleton we write simply "x-zeroing" and/or "from r". For instance, the program from Example 1 has exactly one x-zeroing run from the valuation $\mathsf{x} = 10$, $\mathsf{y} = \mathsf{z} = 0$, where the final values of counters are $\mathsf{x} = 0$, $\mathsf{y} = 10$, $\mathsf{z} = 21$.

By $\mathbf{0}$ we denote the valuation where all counters are 0. Following [9, 10, 11, 12, 31], we investigate the complexity of the following variant of the reachability problem (with a partially specified final valuation of counters):

Reachability problem
Input A program $\mathcal{P}$ without zero tests, and a subset X of its counters.
Question Does $\mathcal{P}$ have an X-zeroing run from the zero valuation $\mathbf{0}$?

Since counter programs without zero tests can be seen as presentations of VASS, the above decision problem translates to a variant of the reachability problem for the latter model, where all components of the initial vector are 0, and the specified components of the final vector are required to be 0. This variant polynomially reduces to the classical one where all components of the final vector are fully specified (say, required to be 0), and the reduction preserves dimension. According to the encoding of VASS as Petri nets, the problem translates to the *submarking reachability* problem for the latter model, where all places (except for those encoding the control states) are initially empty, and the specified places are required to be finally empty. Finally, the submarking reachability problem is polynomially equivalent to a variant where the final content of all places is fully specified.

Fast-growing hierarchy. For a positive integer k, let $\mathbb{N}_k = \{k, 2k, 3k, \ldots\} \subseteq \mathbb{N}$ denote positive multiplicities of k. We define the complexity classes $\mathcal{F}_i$ corresponding to the ith level in the Grzegorczyk Hierarchy [43, Sect. 2.3, 4.1]. The standard family of approximations $\mathbf{A}_i : \mathbb{N}_1 \to \mathbb{N}_1$ of Ackermann function, for $i \in \mathbb{N}_1$, can be defined as follows:

$$\mathbf{A}_1(n) = 2n, \qquad \mathbf{A}_{i+1}(n) = \underbrace{\mathbf{A}_i \circ \mathbf{A}_i \circ \ldots \circ \mathbf{A}_i}_{n}(1) = \mathbf{A}_i^n(1).$$

In particular, $\mathbf{A}_2(n) = 2^n$ and $\mathbf{A}_i(1) = 2$ for all $i \in \mathbb{N}_1$. Using functions $\mathbf{A}_i$, we define the complexity classes $\mathcal{F}_i$, indexed by $i \in \mathbb{N}_1$, of problems solvable in deterministic time $\mathbf{A}_i(p(n))$, where $p : \mathbb{N}_1 \to \mathbb{N}_1$ ranges over functions computable in deterministic time $\mathbf{A}_{i-1}^m(n)$, for some $m \in \mathbb{N}_1$:

$$\mathcal{F}_i = \bigcup_{p \in \mathcal{FF}_{i-1}} \mathrm{DT{\scriptsize IME}}(\mathbf{A}_i(p(n))), \qquad \text{where } \mathcal{FF}_i = \bigcup_{m \in \mathbb{N}_1} \mathrm{FDT{\scriptsize IME}}(\mathbf{A}_i^m(n)).$$

Intuitively speaking, the class $\mathcal{F}_i$ contains all problems solvable in time $\mathbf{A}_i(n)$, and is closed under reductions computable in time of lower order $\mathbf{A}_{i-1}^m(n)$, for some fixed $m \in \mathbb{N}_1$. In particular, $\mathcal{F}_3$ = Tower (problems solvable in a tower of exponentials of time or space, whose height is an elementary function of input size). The classes $\mathcal{F}_k$ are robust with respect to the choice of fast-growing function hierarchy (see [43, Sect.4.1]). For $k \geq 3$, instead of deterministic time, one could equivalently take nondeterministic time, or space.

Dimension-parametric lower bound. As the main result we prove $\mathcal{F}_k$-hardness for programs with the fixed number $3k + 2$ of counters:

▶ **Theorem 2.** *Let $k \geq 3$. The reachabilty problem for programs with $3k + 2$ counters is $\mathcal{F}_k$-hard.*

The proof is in Section 6. The result can be compared to $\mathcal{F}_k$-hardness shown in [12] for $6k$ counters, and in [31] for $4k + 5$ counters. Like the cited results, Theorem 2 implies Ackermann-hardness for unrestricted number of counters which, together with Ackermann upper bound of [34], yields Ackermann-completness of the reachability problem.

Idea of simplification. Czerwiński and Orlikowski [12] use the *ratio technique* introduced previously in [9]. Speaking slightly informally, suppose some three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$ satisfy initially

$$\mathsf{b} = B, \qquad \mathsf{c} > 0, \qquad \mathsf{d} = \mathsf{b} \cdot \mathsf{c}, \tag{1}$$

for some fixed positive integer $B \in \mathbb{N}$. Furthermore, suppose that the initial values of c and d may be arbitrary, in a nondeterministic way, as long as they satisfy the latter equality in (1); they are hence unbounded. Under these assumptions, the ratio technique of [9] allows one to correctly simulate unboundedly many zero tests (in fact, the number of simulated zero tests corresponds to the initial value of c which may be arbitrarily large) on counters bounded by B, at the price of using some auxiliary counters.

As our technical contribution, we improve and simplify the ratio technique. The core idea underlying our simplification is, intuitively speaking, to swap the roles of counters b and c: we observe that the above-defined assumption (1) allows us to correctly simulate exactly $B/2$ zero tests (for B even) on unbounded counters (in fact, on counters bounded by the initial value of c which may be arbitrarily large), without any auxiliary counters. This novel approach is presented in detail in Section 4.

3 Multipliers

Following the lines of [9, 10, 12], we rely on a concept of *multiplier*.

Sets computed by programs. Consider a program $\mathcal{P}$ with counters C, a set of counters $\mathsf{X} \subseteq \mathsf{C}$ and $R \subseteq \mathbb{N}^{\mathsf{C}}$. We define the set X-*computed by $\mathcal{P}$ from R* as the set of all valuations of counters at the end of all X-zeroing (and hence forcedly complete) runs of $\mathcal{P}$ from R. Formally, denoting by $\textsc{Runs}_{\mathcal{P}}(R, \mathsf{X})$ the set of all X-zeroing runs of $\mathcal{P}$ from R, and by $\textsc{fin}(\pi)$ the final counter valuation of a complete run π of $\mathcal{P}$, the set X-computed by $\mathcal{P}$ from R is

$$\textsc{Comp}_{\mathcal{P}}(R, \mathsf{X}) \;=\; \{\textsc{fin}(\pi) \mid \pi \in \textsc{Runs}_{\mathcal{P}}(R, \mathsf{X})\}.$$

We omit X when it is irrelevant. As before, when $\mathsf{X} = \{\mathsf{x}\}$ and/or $R = \{r\}$ are a singleton we write simply 'x-computed' and/or 'from r'.

▶ **Example 3.** The program in Example 1 above, x-computes from the set of all valuations satisfying $\mathsf{y} = \mathsf{z} = 0$ (no constraint for x), the set of all valuations satisfying $\mathsf{x} = 0$ (trivially) and $\mathsf{z} = 2\mathsf{y} + 1$.

Likewise, for a fixed integer $m \in \mathbb{N}$ and a program $\mathcal{P}$ with zero tests, we define the set X-computed by $\mathcal{P}$ from R *using m zero tests*, by restricting the above definition to runs $\pi \in \textsc{Runs}_{\mathcal{P}}(R, \mathsf{X})$ that do exactly m zero tests. This finer variant of the definition will be used in the next section.

Multipliers. Let $\mathsf{b}, \mathsf{c}, \mathsf{d} \in \mathsf{C}$ be some three distinguished counters, and $B \in \mathbb{N}_4$. We define the subset $\textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}) \subseteq \mathbb{N}^{\mathsf{C}}$, called informally the *ratio of B*, consisting of all valuations that satisfy the three conditions (1) and assign 0 to all other counters $\mathsf{x} \in \mathsf{C} \setminus \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$.

▶ **Definition 4.** *A program $\mathcal{M}$ (with no zero tests) with counters C that z-computes from the zero valuation $\mathbf{0}$ the set $\textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$, for some four of its counters $\mathsf{z}, \mathsf{b}, \mathsf{c}, \mathsf{d} \in \mathsf{C}$, we call B-***multiplier***. *In formula:* $\textsc{Comp}_{\mathcal{M}}(\mathbf{0}, \mathsf{z}) = \textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$.

▶ **Example 5.** As a simple example, for every fixed $B \in \mathbb{N}_4$, the following program is a B-multiplier of size $\mathcal{O}(B)$ (several commands are written in one line to save space). Counter z is not used at all.

```
Program $\mathcal{M}_B(\mathsf{b}, \mathsf{c}, \mathsf{d})$:
$\mathsf{b} \mathrel{+}= B$    $\mathsf{d} \mathrel{+}= B$    $\mathsf{c} \mathrel{+}= 1$
loop
    $\mathsf{d} \mathrel{+}= B$    $\mathsf{c} \mathrel{+}= 1$
```

Directly from the definition we derive the following fundamental property of multipliers, to be used in the proofs in Sections 5 and 6:

▷ Claim 6. Let $\mathcal{M}$ be a B-multiplier with counters C as in Definition 4, let $\mathcal{P}$ be a counter program with counters $\mathsf{C} \setminus \{\mathsf{z}\}$, and let $\mathsf{Y} \subseteq \mathsf{C}$. Then the set Y-computed by $\mathcal{P}$ from $\textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$ is equal to the set $(\{\mathsf{z}\} \cup \mathsf{Y})$-computed by the composed program $\mathcal{M}\ \mathcal{P}$ from $\mathbf{0}$:

$$\textsc{Comp}_{\mathcal{P}}(\textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}), \mathsf{Y}) \;=\; \textsc{Comp}_{\mathcal{M}\,\mathcal{P}}(\mathbf{0}, \{\mathsf{z}\} \cup \mathsf{Y}).$$

Proof. The claim is a special case of the following general composition rule: for two programs $\mathcal{P}$ and $\mathcal{Q}$, if $\textsc{Comp}_{\mathcal{P}}(A, \mathsf{X}) = B$ and $\mathcal{Q}$ does not use counters X, then $\textsc{Comp}_{\mathcal{P}\,\mathcal{Q}}(A, \mathsf{X} \cup \mathsf{Y}) = \textsc{Comp}_{\mathcal{Q}}(B, \mathsf{Y})$. Indeed, under the above assumptions $(\mathsf{X} \cup \mathsf{Y})$-zeroing runs of $\mathcal{P}\ \mathcal{Q}$ from A are in mutual correspondence with Y-zeroing runs of $\mathcal{Q}$ from B. ◁

Computing multipliers. For technical convenience we prefer to rely on the following family of functions $\mathbf{F}_i : \mathbb{N}_4 \to \mathbb{N}_4$, indexed by $i \in \mathbb{N}_1$, closely related to functions $\mathbf{A}_i$ (cf. Claim 7 below):

$$\mathbf{F}_1(n) = 2n, \qquad \mathbf{F}_{i+1} = \widetilde{\mathbf{F}_i} \quad \text{where} \quad \widetilde{F}(n) = \underbrace{F \circ F \circ \ldots \circ F}_{n/4}(4). \tag{2}$$

By induction on i one easily shows that $\mathbf{F}_i$ is a linear re-scaling of $\mathbf{A}_i$:

▷ Claim 7. $\mathbf{F}_i(4 \cdot n) = 4 \cdot \mathbf{A}_i(n)$, for $i, n \in \mathbb{N}_1$.

Proof. As $\mathbf{F}_1(n) = 2n$ and $\mathbf{A}_1(n) = 2n$, the claim holds for $i = 1$. Assuming the claim for $i \in \mathbb{N}_1$, by n-fold application thereof we derive the required equality for $i + 1$:

$$\mathbf{F}_{i+1}(4 \cdot n) \;=\; \underbrace{\mathbf{F}_i \circ \ldots \circ \mathbf{F}_i}_{n}(4) \;=\; 4 \cdot \underbrace{\mathbf{A}_i \circ \ldots \circ \mathbf{A}_i}_{n}(1) \;=\; 4 \cdot \mathbf{A}_{i+1}(n). \qquad ◁$$

As a technical core of the proof of Theorem 2, combining our simplification with the lines of [12], we provide an effective construction of B-multipliers with $3k+2$ counters, where $B = \mathbf{F}_k(n)$, of size polynomial in k and n.

▶ **Theorem 8.** *Given $k \in \mathbb{N}_1$ and $n \in \mathbb{N}_4$ one can compute, in time polynomial in k and n, an $\mathbf{F}_k(n)$-multiplier with $3k+2$ counters.*

The proof is in Section 5.

4 Bounded number of zero tests

In this section we provide a novel construction that enables simulating a bounded number m of zero tests (cf. Lemma 12) at the cost of introducing additional 3 counters initialised to the ratio of $B = 2(m+1)$. This construction is a core ingredient of the proofs of Theorems 2 and 8.

Whenever analysing a single run of a program, we denote by $\overline{\mathsf{x}}$ the initial value of a counter x, and by $\underline{\mathsf{x}}$ the final value thereof.

Maximal iteration. In the sequel we intensively use loops of the following form that, intuitively, flush the value from counter f to e, decreasing simultaneously counter d (and possibly execute some further commands):

```
loop                                              (3)
    f −= 1    e += 1    d −= 1    . . .
```

Assuming $\overline{\mathsf{d}} \geq \overline{\mathsf{f}}$, we observe that the amount $\overline{\mathsf{d}} - \underline{\mathsf{d}}$ by which d is decreased as an effect of execution (we use the word *execution* as a synonym to *complete run*) of the above loop may be any value between 0 and $\overline{\mathsf{f}}$. Furthermore, assuming $\overline{\mathsf{d}} \geq \overline{\mathsf{e}} + \overline{\mathsf{f}}$, the equality $\overline{\mathsf{d}} - \underline{\mathsf{d}} = \overline{\mathsf{e}} + \overline{\mathsf{f}}$ holds if and only if

$$\overline{\mathsf{e}} = 0 = \underline{\mathsf{f}}. \tag{4}$$

This simple observation will play a crucial role in the sequel, and deserves a definition:

▶ **Definition 9.** *Whenever an execution of a loop of the form* (3) *satisfies the two equalities* (4) *we call this execution* **maximally iterated***.*

The construction. Let $\mathcal{P}$ be a counter program with counters C, and assume that $\mathcal{P}$ uses zero tests only on two its counters $\mathsf{x}, \mathsf{y} \in \mathsf{C}$ (the construction easily extends to programs with an arbitrary number of zero-tested counters). We add to $\mathcal{P}$ three fresh counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$ (let $\mathsf{C}^* = \mathsf{C} \cup \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$), and transform $\mathcal{P}$ into a program $\mathcal{P}^*$ *without zero tests* that, assuming its initial valuation of counters belongs to $\textsc{Ratio}(2(m+1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}^*)$ for some $m \in \mathbb{N}$, simulates correctly m zero tests (jointly) on counters x, y, as long as their sum is bounded by the initial value of c (cf. Lemma 12).

The transformation proceeds in three steps. First, we accompany every increment (decrement) on x with a decrement (increment) of c, and likewise we do for y:

command	replaced by
x += 1	x += 1 c −= 1
x −= 1	x −= 1 c += 1

command	replaced by
y += 1	y += 1 c −= 1
y −= 1	y −= 1 c += 1

In the resulting program $\overline{\mathcal{P}}$ counters x, y are, intuitively speaking, put on a shared 'budget' c. Assuming x and y initially 0, this clearly enforces $\mathsf{x}+\mathsf{y}$ to not exceed the initial value of c, and the sum $s = \mathsf{c}+\mathsf{x}+\mathsf{y}$ to remain invariant.

As the second step, we replace in $\overline{\mathcal{P}}$ every **zero?** x command by the following macro:

<u>ZERO? x:</u>

```
1: loop
2:     $\mathsf{y} \mathrel{-}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{-}= 1$
3: loop
4:     $\mathsf{c} \mathrel{-}= 1 \quad \mathsf{y} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{-}= 1$
5: loop
6:     $\mathsf{y} \mathrel{-}= 1 \quad \mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{-}= 1$
7: loop
8:     $\mathsf{x} \mathrel{-}= 1 \quad \mathsf{y} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{-}= 1$
9: $\mathsf{b} \mathrel{-}= 2$
```

Likewise we replace every **zero?** y command by an analogous macro ZERO? y obtained from ZERO? x by swapping x and y. This yields the program $\widehat{\mathcal{P}}$ without zero tests. We note that each of the two ZERO? macros preserves the sum $s = \mathsf{c}+\mathsf{x}+\mathsf{y}$, and decrements the counter b by 2. Furthermore, each of the two ZERO? macros decrements d by at most $2s$ (cf. Claim 10 below), and hence the macros preserve the inequality $d \geq \mathsf{b} \cdot s$ (recall that $\mathsf{d} = \mathsf{b} \cdot s$ holds initially).

As the final step we adjoint at the end of $\widehat{\mathcal{P}}$ the following program SET-c-TO-ZERO, thus obtaining the transformed program $\mathcal{P}^*$:

<u>SET-c-TO-ZERO:</u>

```
loop
    $\mathsf{c} \mathrel{-}= 1 \quad \mathsf{d} \mathrel{-}= 2$
ZERO? c
```

The macro ZERO? c is obtained from ZERO? x by swapping x and c. We note that an execution of SET-c-TO-ZERO may decrease the sum $\mathsf{c}+\mathsf{x}+\mathsf{y}$ (but ZERO? c preserves it).

Correctness. Recall that $\widehat{\mathcal{P}}$ preserves the sum $\mathsf{c}+\mathsf{x}+\mathsf{y}$; we denote by s the value of this sum. An execution of ZERO? x is called *maximally iterated* if all four loops are so. Observe that every such execution is forcedly *correct*, i.e. satisfies:

$$\overline{\mathsf{x}} = \underline{\mathsf{x}} = 0, \qquad \overline{\mathsf{y}} = \underline{\mathsf{y}}, \qquad \overline{\mathsf{c}} = \underline{\mathsf{c}}. \tag{5}$$

(Likewise in case of ZERO? y and ZERO? c.) The idea behind ZERO? x is to flush from y to a zero-tested counter x and back, but also flush from c to y and back, in an appropriately nested way that guarantees that the amount $\overline{\mathsf{d}} - \underline{\mathsf{d}}$ by which d is decreased equals $2s$ exactly in maximally iterated executions:

▷ Claim 10. Consider an execution of ZERO? x (resp. ZERO? y) macro, assuming $\overline{\mathsf{d}} \geq 2s$. Then $0 \leq \overline{\mathsf{d}} - \underline{\mathsf{d}} \leq 2s$. Furthermore, the equality $\overline{\mathsf{d}} - \underline{\mathsf{d}} = 2s$ holds if and only if the execution is maximally iterated.

Proof. Consider an execution of ZERO? x, assuming $\overline{\mathsf{d}} \geq 2s$, and let $\dot{\mathsf{y}}$ denote the value of y at the exit from the first loop. The amount by which d is decreased in the two loops in lines 1–2 and 7–8 is at most

$$\Delta_1 = 2(\overline{\mathsf{y}} - \dot{\mathsf{y}}) + \overline{\mathsf{x}}.$$

Furthermore, the amount by which d is decreased in the two loops in lines 3–6 is at most

$$\Delta_2 = 2\bar{\mathsf{c}} + \dot{\mathsf{y}}.$$

The sum $\Delta_1 + \Delta_2$ clearly satisfies $\Delta_1 + \Delta_2 \leq 2s = 2(\bar{\mathsf{c}} + \bar{\mathsf{x}} + \bar{\mathsf{y}})$. It equals $2s$ if and only if $\Delta_1 = 2\bar{\mathsf{y}}$ and $\Delta_2 = 2\bar{\mathsf{c}}$, i.e., exactly when all four loops are maximally iterated. ◁

In consequence, as b is decreased by 2, if the invariant $\mathsf{d} = \mathsf{b} \cdot s$ is preserved by an execution of Zero? x (resp. Zero? y) then the zero test is forcedly correct. Furthermore notice that once the invariant is violated, i.e., $\mathsf{d} > \mathsf{b} \cdot s$, due to the first part of Claim 10 the invariant can not be recovered later. These observations lead to the correctness claim stated in Lemma 12.

In the proof of Lemma 12 we will also need the following corollary of Claim 10, where s denotes, as before, the sum $\mathsf{c} + \mathsf{x} + \mathsf{y}$ at the start of Set-c-to-zero(c):

▷ Claim 11. Consider an execution of Set-c-to-zero(c), assuming $\overline{\mathsf{d}} \geq 2s$. Then $0 \leq \overline{\mathsf{d}} - \underline{\mathsf{d}} \leq 2s$. Furthermore, the equality $\overline{\mathsf{d}} - \underline{\mathsf{d}} = 2s$ holds if and only if the Zero? c macro is maximally iterated.

Proof. Consider an execution of Set-c-to-zero(c) and denote by $\dot{s}$ the value of $\mathsf{c} + \mathsf{x} + \mathsf{y}$ just before entering Zero? c. Thus d decrease by $2(s - \dot{s})$ before entering Zero? c. Moreover, due to Claim 10, the macro Zero? c decreases d by at most $2\dot{s}$, and furthermore the macro decreases d by exactly $2\dot{s}$ if and only if it is maximally iterated. These observations imply the claim. ◁

Recall that $\mathsf{C}^* = \mathsf{C} \cup \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$. We define the C^*-*extension* of a counter valuation $v \in \mathbb{N}^{\mathsf{C}}$ as the extension of v where b, c and d are all set to 0. The C^*-extension of a set $R \subseteq \mathbb{N}^{\mathsf{C}}$ is defined as the set of C^*-extensions of all valuations in R.

▶ **Lemma 12.** *The following sets are equal (as subsets of* $\mathbb{N}^{\mathsf{C}^*}$*):*

- *the* C^**-extension of the set computed by* $\mathcal{P}$ *from* $\mathbf{0}$ *using* m *zero tests.*
- *the set* d*-computed by* $\mathcal{P}^*$ *from* Ratio$(2(m+1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}^*)$.

Proof. For the inclusion of the former set in the latter, we show that for each complete run π of $\mathcal{P}$ from $\mathbf{0}$ that does m zero tests on x, y, there is a corresponding d-zeroing run of $\mathcal{P}^*$ from Ratio$(2(m+1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}^*)$, for any initial value $\bar{\mathsf{c}}$ at least as large as the maximal value of the sum $\mathsf{x} + \mathsf{y}$ along π. The run iterates maximally Zero? x and Zero? y macros, decrements c to 0 in line 2 in Set-c-to-zero(c), and then iterates maximally Zero? c. Thus the final counter valuation of the run is the C^*-extension of the final counter valuation of π.

For the converse direction, consider a d-zeroing run π of $\mathcal{P}^*$ from Ratio$(2(m+1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}^*)$. The initial counter valuation satisfies the equalities $\mathsf{b} = 2(m+1)$ and $\mathsf{d} = 2(m+1) \cdot s$. Each execution of Zero? x or Zero? y or Set-c-to-zero(c) decreases b by 2, and d by at most $2s$ (by the first part of Claims 10 and Claim 11). Therefore, since b and d are not affected elsewhere and $\underline{\mathsf{d}} = 0$ finally, we deduce:

▷ Claim 13. Each execution of Zero? x, Zero? y or Zero? c in π decreases d by *exactly* $2s$.

▷ Claim 14. There are exactly m executions of Zero? x or Zero? y in π.

▷ Claim 15. Finally, $\underline{\mathsf{b}} = 0$.

By Claim 13 and the second part of Claim 10 we derive:

▷ Claim 16. Each execution of Zero? x in π is correct, i.e. satisfies the equalities (5). Likewise for Zero? y.

Analogously, by Claim 13 and the second part of Claim 11 we derive:

▷ Claim 17. Finally, $\underline{\mathsf{c}} = 0$.

Due to Claims 14 and 16, once we project away from π the counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$, we obtain a complete run of $\mathcal{P}$ from $\mathbf{0}$ that does exactly m zero tests, as required. Finally, due to Claims 15 and 17, the C^*-extension of the final counter valuation of the obtained run is exactly the final counter valuation of π. ◀

5 Computing a large multiplier (Proof of Theorem 8)

The proof proceeds by combining the concept of amplifier lifting of [12] with the program transformation of Section 4.

Amplifiers. Let $F : \mathbb{N}_4 \to \mathbb{N}_4$ be a monotone function satisfying $F(n) \geq n$ for $n \in \mathbb{N}_4$. Informally speaking, an *F-amplifier* is a program without zero tests that computes the ratio of $F(B)$ from the ratio of B, for every $B \in \mathbb{N}_4$.

▶ **Definition 18.** *Consider a program $\mathcal{P}$ with counters C without zero tests and distinguished three* input *counters $\mathsf{b}, \mathsf{c}, \mathsf{d} \in \mathsf{C}$ and three* output *counters $\mathsf{b}', \mathsf{c}', \mathsf{d}' \in \mathsf{C}$. The program is called **F-amplifier** if for every $B \in \mathbb{N}_4$, it d-computes from $\textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$ the set $\textsc{Ratio}(F(B), \mathsf{b}', \mathsf{c}', \mathsf{d}', \mathsf{C})$.*

We note that no condition is imposed on d-zeroing runs from counter valuations not belonging to any set $\textsc{Ratio}(B, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$. As an example, consider the following program $\mathcal{L}_\ell$, for $\ell \in \mathbb{N}_1$, with input counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$ and output counters $\mathsf{b}', \mathsf{c}', \mathsf{d}'$:

```
Program $\mathcal{L}_\ell(\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{b}', \mathsf{c}', \mathsf{d}')$:
1: loop
2:     loop
3:         $\mathsf{c} \mathrel{-}= 1$    $\mathsf{c}' \mathrel{+}= 1$    $\mathsf{d} \mathrel{-}= 1$    $\mathsf{d}' \mathrel{+}= \ell$
4:     loop
5:         $\mathsf{c}' \mathrel{-}= 1$    $\mathsf{c} \mathrel{+}= 1$    $\mathsf{d} \mathrel{-}= 1$    $\mathsf{d}' \mathrel{+}= \ell$
6:     $\mathsf{b} \mathrel{-}= 2$    $\mathsf{b}' \mathrel{+}= 2\ell$
7: loop
8:     $\mathsf{c} \mathrel{-}= 1$    $\mathsf{c}' \mathrel{+}= 1$    $\mathsf{d} \mathrel{-}= 2$    $\mathsf{d}' \mathrel{+}= 2\ell$
9: $\mathsf{b} \mathrel{-}= 2$    $\mathsf{b}' \mathrel{+}= 2\ell$
```

▷ Claim 19. The above program is an L_ℓ-amplifier, where $L_\ell : \mathbb{N}_4 \to \mathbb{N}_4 = (x \mapsto \ell \cdot x)$.

Proof sketch. Writing counter valuations as vectors $(\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{b}', \mathsf{c}', \mathsf{d}')$, one shows that the program d-computes, from the set containing just one counter valuation $(B, c, d, 0, 0, 0)$, the set containing one counter valuation $(0, 0, 0, \ell \cdot B, c, \ell \cdot d)$. Indeed, as $\mathsf{d} = 0$ finally, each of the two inner loops in lines 2–5, as well as the last loop in lines 7–8, is forcedly maximally iterated. ◁

Putting $\ell = 1$ we get an identity-amplifier $\mathcal{L}_1(\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{b}', \mathsf{c}', \mathsf{d}')$.

Amplifier lifting. Recall the definition (2) of functions $\mathbf{F}_i$; in particular $\mathbf{F}_1 = L_2$. Let $\mathcal{P}$ be a program with counters C, without zero tests, with distinguished input counters $\mathsf{b}_1, \mathsf{c}_1, \mathsf{d}_1 \in \mathsf{C}$ and output counters $\mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2 \in \mathsf{C}$. We describe a transformation of the program $\mathcal{P}$ to a program $\widetilde{\mathcal{P}}$, also without zero tests, such that assuming that $\mathcal{P}$ is an F-amplifier for some function $F : \mathbb{N}_4 \to \mathbb{N}_4$, the program $\widetilde{\mathcal{P}}$ is an $\widetilde{F}$-amplifier. The program $\widetilde{\mathcal{P}}$ uses, except for the counters of $\mathcal{P}$, three fresh counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$. Thus counters of $\widetilde{\mathcal{P}}$ are $\mathsf{C}^* = \mathsf{C} \cup \{\mathsf{b}, \mathsf{c}, \mathsf{d}\}$. We let input counters of $\widetilde{\mathcal{P}}$ be $\mathsf{b}, \mathsf{c}, \mathsf{d}$, and its output counters be $\mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2$.

In the transformation we use the identity-amplifier $\mathcal{L} = \mathcal{L}_1(\mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2, \mathsf{b}_1, \mathsf{c}_1, \mathsf{d}_1)$ with input counters $\mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2$ and output counters $\mathsf{b}_1, \mathsf{c}_1, \mathsf{d}_1$, and the 4-multiplier $\mathcal{M} = \mathcal{M}_4(\mathsf{b}_1, \mathsf{c}_1, \mathsf{d}_1)$ of Example 5, both without zero tests. For defining the program $\widetilde{\mathcal{P}}$ we apply the transformation of Section 4 (with counters d_1 and d_2 in place of x and y) to the following program $\mathcal{Q}$ built using $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{M}$:

Program $\mathcal{Q}$:

```
$\mathcal{M}$
loop
    $\mathcal{P}$
    zero? $\mathsf{d}_1$
    $\mathcal{L}$
    zero? $\mathsf{d}_2$
$\mathcal{P}$
zero? $\mathsf{d}_1$
```

Program $\widetilde{\mathcal{P}}$:

```
$\overline{\mathcal{M}}$
loop
    $\overline{\mathcal{P}}$
    ZERO? $\mathsf{d}_1$
    $\overline{\mathcal{L}}$
    ZERO? $\mathsf{d}_2$
$\overline{\mathcal{P}}$
ZERO? $\mathsf{d}_1$
SET-c-TO-ZERO
```

Formally, $\widetilde{\mathcal{P}} = \mathcal{Q}^*$. Intuitively speaking, the program $\mathcal{Q}$ directly implements the computation of $\widetilde{F}$ according to the definition: with $2\ell + 1$ zero tests it computes, from $\mathbf{0}$, the ratio of $F^{\ell+1}(4)$. Note that counters of $\mathcal{Q}$ are C while counters of $\widetilde{\mathcal{P}}$ are C^*. Lemma 20 states the crucial amplifier-lifting property of the program transformation $\mathcal{P} \mapsto \widetilde{\mathcal{P}}$.

▶ **Lemma 20.** *If $\mathcal{P}$ is an F-amplifier, then $\widetilde{\mathcal{P}}$ is an $\widetilde{F}$-amplifier.*

Proof. Let $\mathcal{P}$ be an F-amplifier. Thus for every $B \in \mathbb{N}_4$, $\textsc{Comp}_{\mathcal{P}}(\textsc{Ratio}(B, \mathsf{b}_1, \mathsf{c}_1, \mathsf{d}_1, \mathsf{C}), \mathsf{d}_1) = \textsc{Ratio}(F(B), \mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2, \mathsf{C})$. The program $\mathcal{L}$, being an identity-amplifier, d_2-computes from $\textsc{Ratio}(B, \mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2, \mathsf{C})$ the set $\textsc{Ratio}(B, \mathsf{b}_1, \mathsf{c}_1, \mathsf{d}_1, \mathsf{C})$. Let $B = 4(\ell + 1) \in \mathbb{N}_4$ for an arbitrary $\ell \in \mathbb{N}$. As $\mathcal{P}$ is an F-amplifier and $\mathcal{L}$ is an identity-amplifier, we deduce:

▷ Claim 21. $\mathcal{Q}$ computes from $\mathbf{0}$ using $2\ell + 1$ zero tests the set $\textsc{Ratio}(F^{\ell+1}(4), \mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2, \mathsf{C})$.

As $\widetilde{\mathcal{P}} = \mathcal{Q}^*$, by Lemma 12 we deduce:

▷ Claim 22. $\textsc{Comp}_{\widetilde{\mathcal{P}}}(\textsc{Ratio}(4(\ell + 1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C}^*), \mathsf{d}) = \textsc{Ratio}(F^{\ell+1}(4), \mathsf{b}_2, \mathsf{c}_2, \mathsf{d}_2, \mathsf{C})$.

As $B \in \mathbb{N}_4$ was chosen arbitrarily and $\widetilde{F}(B) = F^{\ell+1}(4)$, the last claim says that $\widetilde{\mathcal{P}}$ is an $\widetilde{F}$-amplifier. ◀

▶ Remark 23. The program $\mathcal{P}$ appears twice in the body of $\widetilde{\mathcal{P}}$. This doubling can be easily avoided by re-structuring the loop using explicit **goto** commands. In this way, the size of $\widetilde{\mathcal{P}}$ becomes larger than the size of $\mathcal{P}$ only by a constant.

Proof of Theorem 8. We rely on Lemma 20. Given $k \in \mathbb{N}_1$ and $n \in \mathbb{N}_4$ we compute, in time linear in k, the $\mathbf{F}_k$-amplifier $\mathcal{A}_k$ with $3k + 3$ counters C, by $(k - 1)$-fold application of the amplifier lifting transformation $\mathcal{P} \mapsto \widetilde{\mathcal{P}}$ described above, starting from the $\mathbf{F}_1$-amplifier

$\mathcal{L}_2$ of Claim 19. The construction is linear in k due to Remark 23. Let $\mathsf{b}, \mathsf{c}, \mathsf{d} \in \mathsf{C}$ be input counters of $\mathcal{A}_k$. Relying on Claim 6 in Section 3, the $\mathbf{F}_k(n)$-multiplier is obtained by pre-composing $\mathcal{A}_k$ with an n-multiplier (e.g. $\mathcal{M}_n(\mathsf{b}, \mathsf{c}, \mathsf{d})$ from Section 2) that outputs the set $\textsc{Ratio}(n, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$. The whole construction is thus linear in n.

Finally we observe that the counter b is bounded by n and hence can be eliminated: we encode its values in control locations, by cloning the program into $n+1$ copies, where ith (for $i = 0, \ldots, n$) copy corresponds to the value $\mathsf{b} = i$. The resulting program has $3k+2$ counters. ◀

6 Hardness of the reachability problem (Proof of Theorem 2)

Relying on Lemma 12 and Theorem 8, we prove in this section Theorem 2. Fix $k \geq 3$. The proof proceeds by a polynomial-time reduction from the following $\mathcal{F}_k$-hard problem:

$\mathbf{F}_k$-BOUNDED HALTING PROBLEM
Input A program $\mathcal{P}$ of size n (w.l.o.g. assume $n \in \mathbb{N}_4$) with 2 zero-tested counters.
Question Does $\mathcal{P}$ have a complete run from $\mathbf{0}$ that does at most $(\mathbf{F}_k(n) - 1)/2$ zero tests?

▷ Claim 24. The above problem is $\mathcal{F}_k$-hard.

Proof. Indeed, the standard $\mathcal{F}_k$-hard halting problem (does a program $\mathcal{P}$ with *arbitrarily many* zero-tested counters $\mathsf{x}_1, \ldots, \mathsf{x}_\ell$ have a complete run that does at most $(\mathbf{F}_k(n) - 1)/2$ *steps*?) reduces polynomially to the above one using the standard simulation of arbitrarily many zero-tested counters by 2 such counters $\mathsf{y}_1, \mathsf{y}_2$. The simulation stores the values of all counters $\mathsf{x}_1, \ldots, \mathsf{x}_\ell$ on one of $\mathsf{y}_1, \mathsf{y}_2$ (e.g., using Gödel encoding), and the simulation of each command involves flushing the value of that counter to the other, followed by the zero test. Thus a bound on time of computation is translated to the same bound on the number of zero tests. ◁

Given $\mathcal{P}$ as above with two counters x, y, we transform it to a counter program $\mathcal{P}'$ with $3k+2$ counters C but without zero tests, such that $\mathcal{P}$ has a complete run from $\mathbf{0}$ that does at most $m = (\mathbf{F}_k(n) - 1)/2$ zero tests if and only if $\mathcal{P}'$ has a $\{\mathsf{d}, \mathsf{z}\}$-zeroing run from $\mathbf{0}$ (for some $\mathsf{d}, \mathsf{z} \in \mathsf{C}$).

First, we post-compose $\mathcal{P}$ with a simple program $\mathcal{L}$ that first decrements x nondeterministically many times, and then zero tests it nondeterministically many times:

```
loop
    $\mathsf{x} -= 1$
loop
    zero? $\mathsf{x}$
```

Thus $\mathcal{P}$ has a complete run that does *at most* m zero tests if and only if the composed program $\mathcal{P}\ \mathcal{L}$ has a complete run that does *exactly* m zero tests. We will apply the transformation of Section 4 to the composed program $\mathcal{P}\ \mathcal{L}$. Let $\mathsf{b}, \mathsf{c}, \mathsf{d}$ be the three counters added in the course of the transformation.

Second, using Theorem 8 we compute a $2(m+1)$-multiplier $\mathcal{M}$ (recall that $2(m+1) = \mathbf{F}_k(n)$) with $3k+2$ counters C that z-computes from $\mathbf{0}$ the set $\textsc{Ratio}(2(m+1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$, for some counter z different than x, y. Thus $\mathsf{z}, \mathsf{b}, \mathsf{c}, \mathsf{d} \in \mathsf{C}$.

Finally, we define $\mathcal{P}'$ as a composition of $\mathcal{M}$ with the transformed program $(\mathcal{P}\ \mathcal{L})^*$, and get the required equivalence:

▷ Claim 25. The following conditions are equivalent:

- $\mathcal{P}$ has a complete run from $\mathbf{0}$ that does at most m zero tests;
- $\mathcal{P}\ \mathcal{L}$ has a complete run from $\mathbf{0}$ that does exactly m zero tests;
- $(\mathcal{P}\ \mathcal{L})^*$ has a d-zeroing run from $\textsc{Ratio}(2(m+1), \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{C})$;
- $\mathcal{P}' = \mathcal{M}\ (\mathcal{P}\ \mathcal{L})^*$ has a $\{\mathsf{z}, \mathsf{d}\}$-zeroing run from $\mathbf{0}$.

The second and the third point are equivalent due to Lemma 12, while the equivalence of the third and the last point follows by Claim 6 in Section 3.

The program $\mathcal{P}'$ has $3k+4$ counters ($3k+2$ counters of $\mathcal{M}$ plus x, y) but, since $k \geq 3$, this number can be decreased back to $3k+2$, by re-using some of $3k-2$ counters from $\mathsf{C}' = \mathsf{C} - \{\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{z}\}$ in place of x, y. The latter equivalence in Claim 25 remains true, as z-zeroing runs of $\mathcal{M}$ from $\mathbf{0}$ are necessarily C'-zeroing too, by the definition of multipliers.

This completes the proof of Theorem 2.

7 Final remarks

Primarily, we propose a conceptual simplification of the Ackermann-hardness construction of [12].

As a secondary achievement, we improve the dimension-parametric lower bound for the VASS (Petri nets) reachability problem: compared to $\mathcal{F}_k$-hardness in dimension $6k$ [12] and $4k+5$ [31], respectively, we obtain $\mathcal{F}_k$-hardness already in dimension $3k+2$. (We believe that by combining with the insights of [12] one can further optimise our construction and lower the dimension by a small constant.) The dimension $4k+5$ of [31] has been recently further improved to $2k+4$ [32], thus beating ours.

The best known upper bound places the VASS reachability problem in dimension $k-4$ is in $\mathcal{F}_k$ [34]. Establishing exact parametric complexity of the problem, i.e., closing the gap between dimensions $k-4$ and $2k+4$, arises therefore as an intriguing open problem.

Finally we remind that except for dimension 1 and 2, where the reachability problem seems to be well understood [3, 15], we know no additional complexity bounds for small fixed dimensions k except for the lower bound derived from dimension 2, and the generic $\mathcal{F}_{k+4}$ upper bound of [34].

References

1 David Angeli, Patrick De Leenheer, and Eduardo D. Sontag. Persistence results for chemical reaction networks with time-dependent kinetics and no global conservation laws. *SIAM Journal of Applied Mathematics*, 71(1):128–146, 2011.

2 Paolo Baldan, Nicoletta Cocco, Andrea Marin, and Marta Simeoni. Petri nets for modelling metabolic pathways: a survey. *Natural Computing*, 9(4):955–989, 2010. doi:10.1007/s11047-010-9180-6.

3 Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *Proc. LICS*, pages 32–43, 2015.

4 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27:1–27:26, 2011. doi:10.1145/1970398.1970403.

5 Ahmed Bouajjani and Michael Emmi. Analysis of recursively parallel programs. *ACM Trans. Program. Lang. Syst.*, 35(3):10:1–10:49, 2013. doi:10.1145/2518188.

6 Frank P. Burns, Albert Koelmans, and Alexandre Yakovlev. WCET analysis of superscalar processors using simulation with coloured Petri nets. *Real-Time Systems*, 18(2/3):275–288, 2000. doi:10.1023/A:1008101416758.

7 Thomas Colcombet and Amaldev Manuel. Generalized data automata and fixpoint logic. In *FSTTCS*, volume 29 of *LIPIcs*, pages 267–278. Schloss Dagstuhl, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.267.

8 Stefano Crespi-Reghizzi and Dino Mandrioli. Petri nets and Szilard languages. *Information and Control*, 33(2):177–192, 1977. doi:10.1016/S0019-9958(77)90558-7.

9 Wojciech Czerwiński, Sławomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. In Moses Charikar and Edith Cohen, editors, *Proc. STOC 2019*, pages 24–33. ACM, 2019.

10 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. *J. ACM*, 68(1):7:1–7:28, 2021. doi:10.1145/3422822.

11 Wojciech Czerwinski, Slawomir Lasota, and Łukasz Orlikowski. Improved lower bounds for reachability in vector addition systems. In *Proc. ICALP*, volume 198 of *LIPIcs*, pages 128:1–128:15, 2021.

12 Wojciech Czerwiński and Łukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *Proc. FOCS 2021*, 2021. To appear.

13 Normann Decker, Peter Habermehl, Martin Leucker, and Daniel Thoma. Ordered navigation on multi-attributed data words. In *Proc. CONCUR*, volume 8704 of *LNCS*, pages 497–511. Springer, 2014. doi:10.1007/978-3-662-44584-6_34.

14 Stéphane Demri, Diego Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. *Logical Methods in Computer Science*, 12(3), 2016. doi:10.2168/LMCS-12(3:1)2016.

15 Matthias Englert, Ranko Lazić, and Patrick Totzke. Reachability in two-dimensional unary vector addition systems with states is NL-complete. In *Proc. LICS*, pages 477–484. ACM, 2016. doi:10.1145/2933575.2933577.

16 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Inf.*, 54(2):191–215, 2017. doi:10.1007/s00236-016-0272-3.

17 Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Trans. Program. Lang. Syst.*, 34(1):6:1–6:48, 2012. doi:10.1145/2160910.2160915.

18 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992. doi:10.1145/146637.146681.

19 Sheila A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theor. Comput. Sci.*, 7:311–324, 1978. doi:10.1016/0304-3975(78)90020-8.

20 Piotr Hofman and Sławomir Lasota. Linear equations with ordered data. In *Proc. CONCUR*, volume 118 of *LIPIcs*, pages 24:1–24:17. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.CONCUR.2018.24.

21 John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979. doi:10.1016/0304-3975(79)90041-0.

22 Alexander Kaiser, Daniel Kroening, and Thomas Wahl. A widening approach to multithreaded program verification. *ACM Trans. Program. Lang. Syst.*, 36(4):14:1–14:29, 2014. doi:10.1145/2629608.

23 Max I. Kanovich. Petri nets, Horn programs, linear logic and vector games. *Ann. Pure Appl. Logic*, 75(1–2):107–135, 1995. doi:10.1016/0168-0072(94)00060-G.

24 Richard M. Karp and Raymond E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969. doi:10.1016/S0022-0000(69)80011-5.

25 S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proc. STOC*, pages 267–281. ACM, 1982. doi:10.1145/800070.802201.

26 Jean-Luc Lambert. A structure to decide reachability in Petri nets. *Theor. Comput. Sci.*, 99(1):79–104, 1992. doi:10.1016/0304-3975(92)90173-D.

27 Hélène Leroux, David Andreu, and Karen Godary-Dejean. Handling exceptions in Petri net-based digital architecture: From formalism to implementation on FPGAs. *IEEE Trans. Industrial Informatics*, 11(4):897–906, 2015.

28 Jérôme Leroux. The general vector addition system reachability problem by Presburger inductive invariants. *Logical Methods in Computer Science*, 6(3), 2010. `doi:10.2168/LMCS-6(3:22)2010`.

29 Jérôme Leroux. Vector addition system reachability problem: a short self-contained proof. In *POPL*, pages 307–316. ACM, 2011. `doi:10.1145/1926385.1926421`.

30 Jérôme Leroux. Vector addition systems reachability problem (A simpler solution). In *Turing-100*, volume 10 of *EPiC Series in Computing*, pages 214–228. EasyChair, 2012. URL: `http://www.easychair.org/publications/paper/106497`, `doi:10.29007/bnx2`.

31 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *Proc. FOCS 2021*, 2021. To appear.

32 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive, 2021. `arXiv:2104.12695`.

33 Jérôme Leroux and Sylvain Schmitz. Demystifying reachability in vector addition systems. In *Proc. LICS*, pages 56–67, 2015.

34 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *Proc. LICS*, pages 1–13. IEEE, 2019.

35 Yuliang Li, Alin Deutsch, and Victor Vianu. VERIFAS: A practical verifier for artifact systems. *PVLDB*, 11(3):283–296, 2017. `doi:10.14778/3157794.3157798`.

36 Richard J. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976. URL: `http://cpsc.yale.edu/sites/default/files/files/tr63.pdf`.

37 Martin H. Löb and Stanley S. Wainer. Hierarchies of number-theoretic functions. I. *Archiv für mathematische Logik und Grundlagenforschung*, 13(1–2):39–51, 1970. `doi:10.1007/BF01967649`.

38 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *STOC*, pages 238–246. ACM, 1981. `doi:10.1145/800076.802477`.

39 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984. `doi:10.1137/0213029`.

40 Roland Meyer. A theory of structural stationarity in the *pi*-calculus. *Acta Inf.*, 46(2):87–137, 2009. `doi:10.1007/s00236-009-0091-x`.

41 Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg, 1962. URL: `http://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/`.

42 George S. Sacerdote and Richard L. Tenney. The decidability of the reachability problem for vector addition systems (preliminary version). In *STOC*, pages 61–76. ACM, 1977. `doi:10.1145/800105.803396`.

43 Sylvain Schmitz. Complexity hierarchies beyond elementary. *TOCT*, 8(1):3:1–3:36, 2016.

44 Sylvain Schmitz. The complexity of reachability in vector addition systems. *SIGLOG News*, 3(1):4–21, 2016.

45 Wil M. P. van der Aalst. Business process management as the “killer app” for Petri nets. *Software and System Modeling*, 14(2):685–691, 2015. `doi:10.1007/s10270-014-0424-2`.

Scheduling with Communication Delay in Near-Linear Time

Quanquan C. Liu ✉ 🏠
MIT, CSAIL, Cambridge, MA, US

Manish Purohit ✉ 🏠
Google Research, Mountain View, CA, USA

Zoya Svitkina ✉ 🏠
Google Research, Mountain View, CA, USA

Erik Vee ✉ 🏠
Google Research, Mountain View, CA, uSA

Joshua R. Wang ✉ 🏠
Google Research, Mountain View, CA, USA

Abstract

We consider the problem of efficiently scheduling jobs with precedence constraints on a set of identical machines in the presence of a uniform communication delay. Such precedence-constrained jobs can be modeled as a directed acyclic graph, $G = (V, E)$. In this setting, if two precedence-constrained jobs u and v, with v dependent on u ($u \prec v$), are scheduled on different machines, then v must start at least ρ time units after u completes. The scheduling objective is to minimize makespan, i.e. the total time from when the first job starts to when the last job finishes. The focus of this paper is to provide an efficient approximation algorithm with near-linear running time. We build on the algorithm of Lepere and Rapine [STACS 2002] for this problem to give an $O\left(\frac{\ln \rho}{\ln \ln \rho}\right)$-approximation algorithm that runs in $\tilde{O}(|V| + |E|)$ time.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases near-linear time scheduling, scheduling with duplication, precedence-constrained jobs, graph algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.47

Related Version *Full Version*: `https://arxiv.org/abs/2108.02770` [20]

1 Introduction

The problem of efficiently scheduling a set of jobs over a number of machines is a fundamental optimization problem in computer science that becomes ever more relevant as computational workloads become larger and more complex. Furthermore, in real-world data centers, there exists non-trivial *communication delay* when data is transferred between different machines. There is a variety of very recent literature devoted to the theoretical study of this topic [10, 11, 21]. However, all such literature to date focuses on obtaining algorithms with good approximation factors for the schedule length, but these algorithms require $\omega(n^2)$ time (and potentially polynomially more) to compute the schedule. In this paper, we instead focus on efficient, near-linear time algorithms for scheduling while maintaining an approximation factor equal to that obtained by the best-known algorithm for our setting [19].

Even simplistic formulations of the scheduling problem (e.g. precedence-constrained jobs with unit length to be scheduled on M machines) are typically NP-hard, and there is a rich body of literature on designing good approximation algorithms for the many variations of multiprocessor scheduling (refer to [6] for a comprehensive history of such problems). Motivated by a desire to better understand the computational complexity of scheduling problems and to tackle rapidly growing input sizes, we ask the following research question:

How computationally expensive is it to perform approximately-optimal scheduling?

In this paper, we focus on the classical problem of multiprocessor scheduling with communication delays on identical machines where all jobs have unit size. The jobs that need to be scheduled have data dependencies between them, where the output of one job acts as the input to another. These dependencies are represented using a directed acyclic graph (DAG) $G = (V, E)$ where each vertex $v \in V$ corresponds to a job and an edge $(u, v) \in E$ indicates that job u must be scheduled before v. In our multiprocessor environment, if these two jobs are scheduled on different machines, then some additional time must be spent to transfer data between them. We consider the problem with *uniform communication delay*; in this setting, a uniform delay of ρ is incurred for transferring data between any two machines. Thus for any edge $(u, v) \in E$, if the jobs u and v are scheduled on different machines, then v must be scheduled at least ρ units of time after u finishes. Since the communication delay ρ may be large, it may actually be more efficient for a machine to *recompute* some jobs rather than wait for the results to be communicated. Such duplication of work can reduce schedule length by up to a logarithmic factor [21] and has been shown to be effective in minimizing latency in schedulers for grid computing and cloud environments [5, 7]. Our scheduling objective is to minimize the makespan of the schedule, i.e., the completion time of the last job. In the standard three field notation for scheduling problems, this problem is denoted "$P \mid \text{duplication}, \text{prec}, p_j = 1, c \mid C_{\max}$", where c indicates uniform communication delay.

This problem was studied by Lepere and Rapine, who devised an $O(\ln \rho / \ln \ln \rho)$-approximation algorithm for it [19], under the assumption that the optimal solution takes at least ρ time. However, their analysis was primarily concerned with getting a good quality solution and less with optimizing the running time of their polynomial-time algorithm. A naïve implementation of their algorithm takes roughly $O(m\rho + n \ln M)$ time, where n and m are the numbers of vertices and edges in the DAG, respectively, and M is the number of machines. This runtime is based on two bottlenecks, (i) the computation of ancestor sets, which can be done in $O(m\rho)$ time via propagating in topological order plus merging and (ii) list scheduling, which can be done in $O(n \ln M)$ time by using a priority queue to look up the least loaded machine when scheduling a set of jobs.

However, with growing input sizes, it is highly desirable to obtain a scheduling algorithm whose running time is linear in the size of the input. Our primary contribution is to design a *near-linear time* randomized approximation algorithm while preserving the approximation ratio of the Lepere-Rapine algorithm:

▶ **Theorem 1.** *There is an $O(\ln \rho / \ln \ln \rho)$-approximation algorithm for scheduling jobs with precedence constraints on a set of identical machines in the presence of a uniform communication delay that runs in $O\left(n \ln M + \frac{m \ln^3 n \ln \rho}{\ln \ln \rho}\right)$ time, with high probability, assuming that the optimal solution has cost at least ρ.*

Of course, this is tight, up to log factors, because any algorithm for this problem must respect the precedence constraints, which require $\Omega(n + m)$ time to read in. In the settings where our algorithm is more efficient than Lepere-Rapine, the approximation factor of the algorithm is still very small (near-constant in the cases where $\rho = \operatorname{poly}\log n$), yet our algorithm achieves a better runtime while maintaining the same approximation compared to the previous best-known algorithm for the problem.

1.1 Related Work

Algorithms for scheduling problems under different models have been studied for decades, and there is a rich literature on the topic (refer to [6] for a comprehensive look). Here we review work on theoretical aspects of scheduling with communication delay, which is most relevant to our results.

Without duplication, scheduling a DAG of unit-length jobs with unit communication delay was shown to be NP-hard by Rayward-Smith [29], who also gave a 3-approximation for this problem. Munier and König gave a 4/3-approximation for an unbounded number of machines [24], and Hanen and Munier gave a 7/3-approximation for a bounded number of machines [15]. Hardness of approximation results were shown in [3, 16, 28]. In recent results, Kulkarni et al. [18] gave a quasi-polynomial time approximation scheme for a constant number of machines and a constant communication delay, whereas Davies et al. [10] gave an $O(\log \rho \log M)$ approximation for general delay and number of machines. Even more recently, Davies et al. [11] presented a $O(\log^4 n)$-approximation algorithm for the problem of minimizing the weighted sum of completion times on *related machines* in the presence of communication delays. They also obtained a $O(\log^3 n)$-approximation algorithm under the same model but for the problem of minimizing makespan under communication delay. Notably, *none* of the aforementioned algorithms consider duplication and the most recent algorithms have running times that are large polynomials.

Allowing the duplication of jobs was first studied by Papadimitriou and Yannakakis [27], who obtained a 2-approximation algorithm for scheduling a DAG of identical jobs on an unlimited number of identical machines. A number of papers have improved the results for this setting [1, 9, 26]. With a finite number of machines, Munier and Hanen [23] proposed a 2-approximation algorithm for the case of unit communication delay, and Munier [22] gave a constant approximation for the case of tree precedence graphs. For a general DAG and a fixed delay ρ, Lepere and Rapine [19] gave an algorithm that finds a solution of cost $O(\log \rho / \log \log \rho) \cdot (OPT + \rho)$, which is a true approximation if one assumes that $OPT \geq \rho$. This is the main result that our paper builds on. It applies to a set of identical machines and a set of jobs with unit processing times. Recently, an $O(\log M \log \rho / \log \log \rho)$ approximation has been obtained for a more general setting of M machines that run at different speeds and jobs of different lengths by Maiti et al. [21], also under the assumption that $OPT \geq \rho$. However, the running time of this algorithm is a large polynomial ($\omega(n^2)$), as it requires solving an LP with $\Omega(Mn^2)$ variables.

Our results so far only apply to scheduling with duplication. In Maiti et al. [21], a polynomial-time reduction is presented that transforms a schedule with duplication into one without duplication (with a polylogarithmic increase in makespan). However, this reduction involves constructing an auxiliary graph of possibly $\Omega(\rho^2)$ size, and thus does not lend itself easily to a near-linear time algorithm. It would be interesting to see if a near-linear time reduction could be found.

1.2 Technical Contributions

A naïve implementation of the Lepere-Rapine algorithm is bottlenecked by the need to determine the set of all ancestors of a vertex v in the graph, as well as the intersection of this set with a set of already scheduled vertices. Since the ancestor sets may significantly overlap with each other, trying to compute them explicitly (e.g., using DFS to write them down) results in superlinear work. We use a variety of technical ideas to only compute the essential size information that the algorithm needs to make decisions about these ancestor sets.

- **Size estimation via sketching.** We use streaming techniques to quickly estimate the sizes of all ancestor sets simultaneously. It costs $O((|V| + |E|)\log^2 n)$ time to make such an estimate once, so we are careful to do so sparingly.
- **Work charging argument.** Since we cannot compute our size estimates too often, we still need to perform some DFS for ancestor sets. We control the amount of work spent doing so by carefully charging the edges searched to the edges we manage to schedule.
- **Sampling and pruning.** Because we cannot brute-force search all ancestor sets, we randomly sample vertices, using a consecutive run of unscheduleable vertices as evidence that many vertices are not schedulable. This allows us to pay for an expensive size-estimator computation to prune many ancestor sets simultaneously.

1.3 Organization

The main contribution of this paper is our algorithm for scheduling small subgraphs in near-linear time. We provide a detailed description and analysis of this algorithm in Section 5. Then, we proceed with our algorithm for scheduling general graphs in Section 6. Due to space constraints we defer all proofs of our analysis to the Appendix.

2 Problem Definition and Preliminaries

An instance of scheduling with communication delay is specified by a directed acyclic graph $G = (V, E)$, a quantity $M \geq 1$ of identical machines, and an integer communication delay $\rho > 1$. We assume that time is slotted and let $T = \{1, 2, \ldots\}$ denote the set of integer times. Each vertex $v \in V$ corresponds to a job with processing time 1 and a directed edge $(u, v) \in E$ represents the precedence constraint that job v depends on job u. In total, there are $n = |V|$ vertices (representing jobs) and $m = |E|$ precedence constraints. The parameter ρ indicates the amount of time required to communicate the result of a job computed on one machine to another. In other words, a job v can be scheduled on a machine at time t only if all jobs u with $(u, v) \in E$ have either completed on the same machine before time t or on another machine before time $t - \rho$. We allow for a job to be *duplicated*, i.e., copies of the same vertex $v \in V$ may be processed on different machines. Let $\mathcal{M}$ be the set of machines available to schedule the jobs. A schedule σ is represented by a set of triples $\{(m, v, t)\} \subset \mathcal{M} \times V \times T$ where each triple represents that job v is scheduled on machine m at time t. The goal is to obtain a feasible schedule that minimizes the makespan, i.e., the completion time of the last job. Let OPT denote the makespan of an optimal schedule. Since ρ represents the amount of time required to communicate between machines, and in practice, any schedule must communicate the results of the computation, we assume that $\mathsf{OPT} \geq \rho$ as is standard in literature [19, 21].

We now set up some notation to help us better discuss dependencies arising from the precedence constraints of G. For any vertex $v \in V$, let $\mathsf{Pred}(v) \triangleq \{u \in V \mid (u, v) \in E\}$ be the set of (immediate) predecessors of v in the graph G, and similarly let $\mathsf{Succ}(v) \triangleq \{w \in V \mid (v, w) \in E\}$ be the set of (immediate) successors. For $H = (V_H, E_H)$, a subgraph of G, we use $\mathcal{A}_H(v) \triangleq \{u \in V_H \mid \exists \text{ a directed path from } u \text{ to } v \text{ in } H\} \cup \{v\}$ to denote the set of (indirect) ancestors of v, including v itself. Similarly, for $S \subseteq V$, we use $\mathcal{A}_H(S) \triangleq \bigcup_{v \in S} \mathcal{A}_H(v)$ to denote the indirect ancestors of the entire set S. We use $\mathcal{E}_H(S)$ to denote the edges of the subgraph induced by $\mathcal{A}_H(S)$. We drop the subscript H when the subgraph H is clear from context. Throughout, we use the phrase *with high probability* to indicate with probability at least $1 - \frac{1}{n^c}$ for any constant $c \geq 1$.

For convenience, we summarize the notation we use throughout the paper in Table 1.

Table 1 Table of Symbols.

Symbol	Meaning
$G = (V, E)$	main input graph
$n = \|V\|, m = \|E\|$	number of vertices / edges
$H = (V_H, E_H)$	subgraph to be scheduled in each phase
ρ	communication delay
u, v	vertices
$\mathcal{A}_H(v)$	set of ancestors of vertex v in graph H including v
$\mathcal{A}_H(S)$	$\mathcal{A}_H(S) = \bigcup_{v \in S} \mathcal{A}_H(v)$ in graph H
$\mathcal{E}_H(v)$ [resp., $\mathcal{E}_H(S)$]	edges induced by $\mathcal{A}_H(v)$ [resp., $\mathcal{A}_H(S)$] in graph H
$\hat{a}_H(v), \hat{e}_H(v)$	estimated size of $\mathcal{A}_H(v)$ and $\mathcal{E}_H(v)$
M	number of machines
γ	threshold for fresh vs. stale vertices

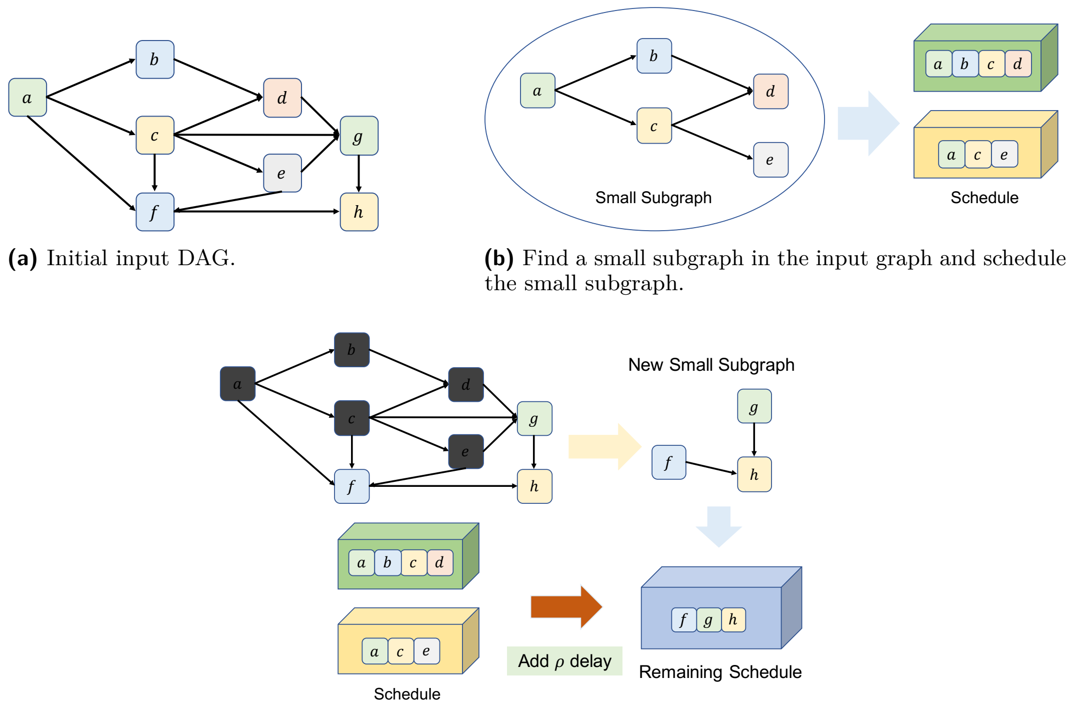
3 Technical Overview

We start by reviewing the algorithm of Lepere and Rapine [19], shown in Algorithm 1, as our algorithm follows a similar outline. Then we describe the technical improvements of our algorithm to achieve near-linear running time.

Algorithm 1 Outline of Lepere Rapine Scheduling Algorithm [19].

```
while $G$ is non-empty do
    Let $H$ be a subgraph of $G$ induced by vertices with at most $\rho + 1$ ancestors
    while $H$ is non-empty do
        for each vertex $v$ in $H$ do
            if greater than $\gamma$ fraction of $\mathcal{A}_H(v)$ is unscheduled then
                Add $\mathcal{A}_H(v)$, in topological order, to a machine with earliest end time
        Insert a delay until $C + \rho$ on all machines, where $C$ is the latest end time
        Remove scheduled vertices from $H$
    Delete vertices in $H$ from $G$
```

Description of Lepere-Rapine [19]. The outer loop (Algorithm 1) iteratively finds *small subgraphs* of G which consist of vertices that have *height* at most $\rho + 1$. We show in this paper that instead of considering their definition of *height*, it is sufficient in our algorithm to consider small subgraphs to be those with at most 2ρ ancestors. We call one iteration of this loop a *phase*. Within the phase, H is fully scheduled, after which the algorithm goes on to the next "slice" of G. However, H is not scheduled all at once, but instead each iteration of the inner `while` loop (Algorithm 1) schedules a subset of H, which we call a *batch*. To determine which vertices of H make it into a batch, the algorithm checks the fraction of ancestors of each vertex that have already been scheduled in the same batch. If this fraction for a vertex v is low (we call v *fresh* in that case), then its ancestor set $\mathcal{A}_H(v)$ is list-scheduled as a unit, i.e. ancestor jobs are duplicated, topologically sorted, and placed on one machine. If the fraction of scheduled ancestors is high (in which case we call v *stale*), v is skipped in this iteration. We skip v to avoid excessive duplication that would create too much load on



(a) Initial input DAG.

(b) Find a small subgraph in the input graph and schedule the small subgraph.

(c) Remove scheduled vertices from the graph. Add a ρ communication delay to the schedule after the previously scheduled jobs. Find a new small subgraph and schedule it.

Figure 1 Overview of the Lepere-Rapine algorithm for scheduling general graphs.

the machines. After each batch is placed on the machines, a delay of ρ is added to the end of the schedule to allow all the results to propagate. This allows the scheduled jobs to be deleted from H. This algorithm is illustrated pictorially in Figure 1.

Runtime Challenges with Lepere-Rapine. Naively, both finding the small subgraphs as well as determining each batch takes $\Omega(n\rho^2)$ time. Determining which nodes belong in the current small subgraph is a matter of whether their ancestor counts are more than ρ or at most ρ. A standard procedure would be to apply DFS and merge ancestor sets, but that can easily run in $\Omega(\rho^2)$ time per node (a node may have $\Omega(\rho)$ direct parents, each with an ancestor size of $\Omega(\rho)$ that needs to get merged in).

The other technical hurdle is in determining the batches to schedule. We would like to schedule vertices whose ancestors do not *overlap too much*. To illustrate the difficulty of applying sketching-based methods (e.g. min-hash), consider the following example. Suppose that ρ^2 elements have already been scheduled in this batch. Now, we want to find the number of ancestors of vertex v, $\mathcal{A}(v)$, that intersect with the currently scheduled batch, where $|\mathcal{A}(v)| \leq \rho$ by construction. By the lower bound given in [25], even estimating (up to $1 \pm \epsilon$ relative error with constant probability) the size of this intersection would require sketches of size at least $\varepsilon^{-2}(\rho^2/\rho) = \varepsilon^{-2}\rho$. Using such ρ-sized sketches over all batches and all small subgraphs require $\Omega(n\rho)$ time in total.

Since ρ may be super-logarithmic, these naive implementations don't quite meet our goal of a near-linear time algorithm. To summarize, the two main technical challenges for our setting are the following:

▶ **Challenge 1.** *We must be able to find the small subgraphs in near-linear time.*

▶ **Challenge 2.** *We must be able to find the vertices to add to each batch B in near-linear time.*

We solve Challenge 1 by relaxing the definition of small subgraph and using *count-distinct* estimators (discussed in Section 4). The majority of our paper focuses on solving Challenge 2 which requires several new techniques for the problem outlined in the rest of this section (Section 3.1 and Section 3.2). The below procedures run on a small subgraph, $H = (V_H, E_H)$, where the number of ancestors of each vertex is bounded by 2ρ. Note the factor of 2 results from our count-distinct estimator. This is described in Section 5. Our algorithm for scheduling small subgraphs is shown pictorially in Figure 2.

3.1 Sampling Vertices to Add to the Batch

We first partition the set of unscheduled vertices in V_H into buckets based on the estimated number of edges in the subgraph induced by their ancestors. (We place vertex v – if it has no ancestors – into the bucket with the smallest index.) We partition by edges instead of vertices because the number of edges in the induced subgraph of the ancestors affects our running time. More formally, let S_i be the set of vertices not yet scheduled in iteration i (Algorithm 1, Algorithm 1). We partition S_i into $k = O(\log \rho)$ buckets $K_1, \dots, K_k$ such that bucket K_j contains all vertices $w \in S_i$ where $\hat{e}(w) \in [2^j, 2^{j+1})$; $\hat{e}(w)$ denotes the estimated number of edges in the subgraph induced by ancestors of w.

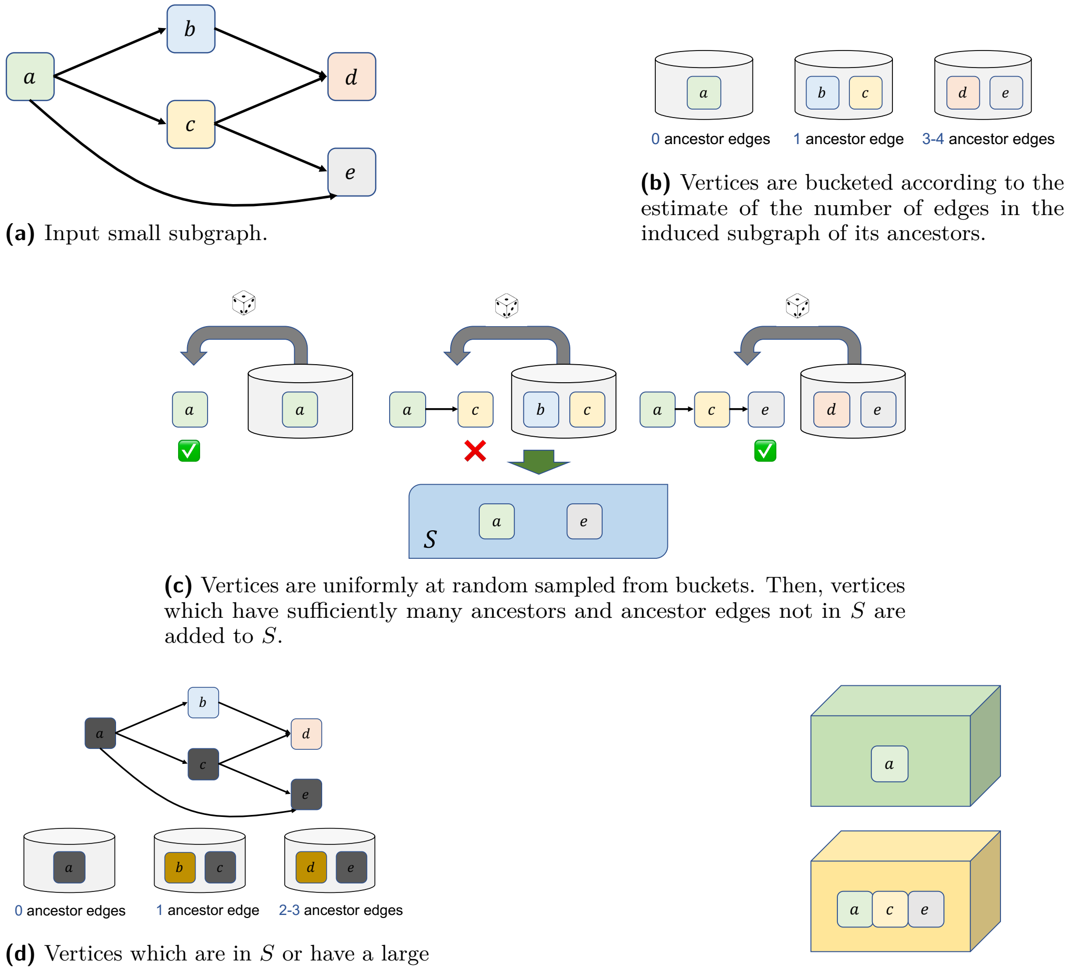
From each bucket K_j, in decreasing order of j, we sample vertices, sequentially, without replacement. For each sampled vertex v, we enumerate its ancestors and determine how many are in the current batch B. If at least a γ-fraction of the vertices *are not in B and* at least a γ-fraction of the edges (with both endpoints in B) in the induced subgraph $G_{H_i}(v)$ are *not in B*, then add v and all its ancestors to B. We call such a vertex v ***fresh***. Otherwise, we do not add v to B and label this vertex as ***stale***. For our algorithms, we set $\gamma = \frac{1}{\sqrt{\rho}}$ to minimize the approximation factor but γ can be set to any value $\gamma < 1/2$. Lepere-Rapine did not consider edges in their algorithm because the number of edges in the induced subgraph does not affect the schedule length; however, considering edges is crucial for our algorithm to run in near-linear time.

For each bucket sequentially, we sample vertices uniformly at random, until we have sampled $O(\log n)$ consecutive vertices that are *stale* (or we have run out of vertices and the bucket is empty). Then, the key intuition is that for every v that we add to B, we can afford to *charge the cost of enumerating the ancestor set* for $O(\log n)$ additional vertices in the same bucket as well as $O(\log n)$ additional vertices in each bucket with smaller j to it. Because we are looking at buckets with decreasing indices, we can charge the additional vertices found in future buckets to the most recently found fresh vertex.

3.2 Pruning All Stale Vertices from Buckets

After we have performed the sampling procedure, we are still not done. Our goal is to make sure that *all vertices which are not included* in B are approximately stale. This means that we must remove the stale vertices so that we can perform our sampling procedure again in a smaller sample space in order to find additional fresh vertices. To accomplish this, we perform a ***pruning*** procedure involving re-estimating the ancestor sets consisting of vertices that have not been added to the batch. Using these estimates, we remove *all* stale vertices from our buckets. Note that we *do not rebucket the vertices* because none of the ancestor sets of the vertices changed sizes. Then, we perform our sampling procedure above (again) to find more fresh vertices. The key is that since we removed all stale vertices, *the first sampled vertex from the non-empty bucket with the largest index is fresh.*

We perform the above sampling and pruning procedures until each bucket is empty. Then, we schedule the batch and remove all scheduled vertices from H and proceed again with the procedure until the graph is empty. We perform a standard simple greedy list scheduling algorithm (Appendix B) on our batch on M machines.



(a) Input small subgraph.

(b) Vertices are bucketed according to the estimate of the number of edges in the induced subgraph of its ancestors.

(c) Vertices are uniformly at random sampled from buckets. Then, vertices which have sufficiently many ancestors and ancestor edges not in S are added to S.

(d) Vertices which are in S or have a large proportion of ancestors or ancestor edges in S are pruned from buckets. b and d are pruned in this example.

(e) Vertices in S and all ancestors are scheduled by duplicating ancestors and list scheduling.

Figure 2 Overview of our scheduling small subgraphs algorithm. We choose $\gamma = 2/3$ here for illustration purposes but in our algorithms $\gamma < 1/2$.

4 Estimating Number of Ancestors

Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be an arbitrary directed, acyclic graph. We first present our algorithm to estimate the number of ancestors of any vertex $v \in \tilde{V}$. Consider any vertex $v \in \tilde{V}$ and let $p_1, p_2, \ldots p_\ell$ be the predecessors of v in $\tilde{G}$. Then we have $\mathcal{A}_{\tilde{G}}(v) = \cup_{i=1}^{\ell} \mathcal{A}_{\tilde{G}}(p_i) \cup \{v\}$ and hence $|\mathcal{A}_{\tilde{G}}(v)|$ is the number of distinct elements in the multiset $\cup_{i=1}^{\ell} \mathcal{A}_{\tilde{G}}(p_i) \cup \{v\}$. In order to estimate $|\mathcal{A}_{\tilde{G}}(v)|$ efficiently, we use a procedure to estimate the number of distinct elements in a data stream. This problem, known as the *count-distinct problem*, is well studied and many efficient estimators exist [2, 4, 30, 12, 17]. Since we need to estimate $|\mathcal{A}_{\tilde{G}}(v)|$ for all vertices $v \in \tilde{V}$ in near-linear time, we require an additional *mergeable* property to ensure that we can efficiently obtain an estimate for $|\mathcal{A}_{\tilde{G}}(v)|$ from the estimates of the parent ancestor set sizes $\{|\mathcal{A}_{\tilde{G}}(p_1)|, \ldots, |\mathcal{A}_{\tilde{G}}(p_\ell)|\}$.

We formally define the notion of a *count-distinct estimator* and the mergeable property.

▶ **Definition 2.** *For any multiset $\mathcal{S}$, let $|\mathcal{S}|$ denote the number of distinct elements in $\mathcal{S}$. We say T is an (ε, δ, D)-*CountDistinctEstimator *for $\mathcal{S}$ if it uses space D and returns a value $\hat{s}$ such that $(1-\varepsilon)|\mathcal{S}| \leq \hat{s} \leq (1+\varepsilon)|\mathcal{S}|$ with probability at least $(1-\delta)$.*

▶ **Definition 3** (Mergeable Property)**.** *An (ε, δ, D)-*CountDistinctEstimator *exhibits the* mergeable property *if estimator T_1 for multiset $\mathcal{S}_1$ and estimator T_2 for multiset $\mathcal{S}_2$ can be merged in $O(D)$ time using $O(D)$ space into an (ε, δ, D)-estimator for $\mathcal{S}_1 \cup \mathcal{S}_2$.*

We note that the *count-distinct estimator* in [4] satisfies the mergeable property and suffices for our purposes. We include a description of the procedure and a proof of the mergeable property in Appendix A.

▶ **Lemma 4** ([4])**.** *For any constant $\varepsilon > 0$ and $d \geq 1$, there exists an $\left(\varepsilon, \frac{1}{n^d}, O\left(\frac{1}{\varepsilon^2}\log^2 n\right)\right)$-*CountDistinctEstimator *that satisfies the mergeable property where n denotes an upper bound on the number of distinct elements.*

Given such an estimator, one can readily estimate the number of ancestors of each vertex $v \in \tilde{V}$ in near-linear time by traversing the vertices of the graph in topological order. An estimator for vertex v can be obtained by *merging* the estimators for each predecessor of v. Similarly, we can also estimate the number of edges $|\mathcal{E}(v)|$ in the subgraph induced by ancestors of any vertex v in near-linear time. We defer a detailed description of these procedures to Algorithm 7 in Appendix A and Algorithm 9 in our full paper [20].

▶ **Lemma 5.** *Given any input graph $\tilde{G} = (\tilde{V}, \tilde{E})$ and constants $\varepsilon > 0, d \geq 1$, there exists an algorithm that runs in $O\left((|\tilde{V}| + |\tilde{E}|)\log^2 n\right)$ time and returns estimates $\hat{a}(v)$ and $\hat{e}(v)$ for each $v \in \tilde{V}$ such that $(1-\varepsilon)|\mathcal{A}_{\tilde{G}}(v)| \leq \hat{a}(v) \leq (1+\varepsilon)|\mathcal{A}_{\tilde{G}}(v)|$* and *$(1-\varepsilon)|\mathcal{E}_{\tilde{G}}(v)| \leq \hat{e}(v) \leq (1+\varepsilon)|\mathcal{E}_{\tilde{G}}(v)|$ with probability at least $1 - \frac{1}{n^d}$.*

Proof. Lemma 15 provides us with our desired approximation. Now, all that remains to show is that Algorithm 7 and Algorithm 9 in our full paper [20] runs within our desired time bounds. Algorithm 7 visits each vertex exactly once. For each vertex, it merges the estimators of each of its immediate predecessors. By Lemma 16, each merge takes $O\left(\frac{1}{\varepsilon^2}\log^2 n\right)$ time. Because we visit each vertex exactly once, we also visit each predecessor edge exactly once. This means that in total we perform $O\left(\frac{m}{\varepsilon^2}\log^2 n\right)$ merges. Since ε is constant, this algorithm requires $O(m \log^2 n)$ time. The same proof follows for Algorithm 9 in our full paper [20]. ◀

Throughout the remaining parts of the paper, we assume that $\varepsilon = 1/3$ in our estimation procedures and do not explicitly give our results in terms of ε.

5 Scheduling Small Subgraphs in Near-Linear Time

Here, we consider subgraphs $H = (V, E)$ such that every vertex in the graph has a bounded number of ancestors and obtain a schedule for such *small subgraphs* in near-linear time.

▶ **Definition 6.** *A* small subgraph *is a graph $H = (V_H, E_H)$ where each vertex $v \in V_H$ has at most 2ρ ancestors.*

Our main algorithm schedules a small graph in batches using Algorithm 3. After scheduling a batch of vertices, we insert a communication delay of ρ time units so that results of the computation from the previous batch are shared with all machines (similar to Lepere-Rapine). Then, we remove all vertices that we scheduled and compute the next batch from the smaller graph. We present this algorithm in Algorithm 2.

Algorithm 2 ScheduleSmallSubgraph(H, γ).

```
Result: A schedule of small subgraph $H$ on $M$ processors.
Input: $H = (V_H, E_H)$ where $|\mathcal{A}_H(v)| \leq 2\rho$ for all $v \in V_H$ and parameter
       $0 < \gamma < 1/2$.
while $H \neq \varnothing$ do
    $B \leftarrow$ FindBatch$(H, \gamma)$. [Algorithm 3]
    List schedule $\mathcal{A}(v)$ using the $M$ processors for all $v \in B$. (Appendix B)
    Insert communication delay of $\rho$ time units into the schedule.
    Remove each $v \in \mathcal{A}(B)$ and all edges adjacent to $v$ from $H$.
end
```

Our algorithm for scheduling small subgraphs relies on two key building blocks – estimating the sizes of the ancestor sets (and ancestor edges) of each vertex (Section 4), and using these estimates to find a *batch* of vertices that can be scheduled without any communication (possibly by duplicating some vertices). We show how to find a batch in Section 5.1.

5.1 Batching Algorithm

Recall that the plan is for our algorithm to schedule a small subgraph by successively scheduling maximal subsets of vertices in the graph whose ancestors do not *overlap too much*; we call such a set of vertices a ***batch***. After scheduling each batch, we remove all the scheduled vertices from the graph and iterate on the remaining subgraph.

A detailed description of this procedure is given in Algorithm 3. For each vertex $v \in V_H$, let $\hat{a}(v)$ and $\hat{e}(v)$ denote the estimated sizes of $\mathcal{A}_H(v)$ and $\mathcal{E}_H(v)$ respectively (henceforth referred to as $\mathcal{A}(v)$ and $\mathcal{E}(v)$). Then the i-th bucket, K_i, is defined as $K_i = \{v \in V_H \mid 2^i \leq \hat{e}(v) < 2^{i+1}\}$. Since every node $v \in V_H$ has at most $O(\rho)$ ancestors, there are only $k = O(\log \rho)$ such buckets. Recall that from Lemma 5, this estimation can be performed in near-linear time. The algorithm maintains a batch B of vertices that is initially empty. For each non-empty bucket K_i (processed in decreasing order of size), we repeatedly sample nodes uniformly at random from the bucket (without replacement).

For each sampled node $v \in K_i$, we explicitly enumerate the ancestor sets $\mathcal{A}(v)$ and $\mathcal{E}(v)$ and also compute $\mathcal{A}(v) \setminus \mathcal{A}(B)$ and $\mathcal{E}(v) \setminus \mathcal{E}(B)$. Since we can maintain the ancestor sets of the current batch B in a hash table, this enumeration takes $O(|\mathcal{E}(v)|)$ time. A sampled node v is said to be *fresh* if $|\mathcal{A}(v) \setminus \mathcal{A}(B)| > \gamma|\mathcal{A}(v)|$ *and* $|\mathcal{E}(v) \setminus \mathcal{E}(B)| > \gamma|\mathcal{E}(v)|$; and said to be *stale* otherwise. The algorithm adds all fresh nodes to the batch B and continues sampling from the bucket until it samples $\Theta(\log n)$ consecutive stale nodes. Once all the buckets have been processed, we *prune* the buckets to remove all stale nodes and then repeat the sampling procedure until all buckets are empty.

A bucket K_i is reduced when 1) a vertex, v, in it is added to B, 2) a sampled vertex v is stale and 3) during the pruning process. No vertex remains unscheduled because either a vertex v is scheduled in the current batch or it is stale. For all stale vertices, in Algorithm 2, we remove all the scheduled ancestors of these stale vertices (so the vertices become fresh again). We repeat the procedure given in Algorithm 3 (Algorithm 2 of Algorithm 2) until the entire graph is scheduled (Algorithm 2 of Algorithm 2) so that all vertices are eventually scheduled. The pruning procedure is presented in Algorithm 4. In this step, we again estimate the sizes of ancestor sets of all vertices in the graph $H \setminus \mathcal{A}(B)$ to determine whether a vertex is stale.

Algorithm 3 FindBatch(H, γ).

```
Result: Returns batch $B$, the batch of vertices to schedule.
Input: A subgraph $H = (V_H, E_H)$ such that $|\mathcal{A}_H(v)| \le 2\rho$ for all $v \in V_H$;
       $0 < \gamma < 1/2$.
Let $N = \Theta(\log n)$.
Initially, $B \leftarrow \varnothing$ and all nodes are unmarked.
Obtain estimates $\hat{a}(v)$ and $\hat{e}(v)$ for all $v \in V_H$.
Let bucket $K_i = \{v \in V_H : 2^i \le \hat{e}(v) < 2^{i+1}\}$.
while at least one bucket is non-empty do
    for $i = k$ to 1 do
        Let $s = 0$.
        while $s < N$ and $|K_i| > 0$ do
            Let $v$ be a uniformly sampled node in bucket $K_i$.
            Find $\mathcal{A}(v)$ and $\mathcal{A}(v) \setminus \mathcal{A}(B)$ as well as $\mathcal{E}(v)$ and $\mathcal{E}(v) \setminus \mathcal{E}(B)$.
            if $|\mathcal{A}(v) \setminus \mathcal{A}(B)| > \gamma|\mathcal{A}(v)|$ and $|\mathcal{E}(v) \setminus \mathcal{E}(B)| > \gamma|\mathcal{E}(v)|$ then
                Mark $v$ as fresh, add $v$ and its ancestors to $B$.
                Set $s = 0$.
            else
                Mark $v$ as stale. $s = s + 1$.
            Remove $v$ from $K_i$.
    $K_1, \ldots, K_k \leftarrow$ Prune($H$, $B$, $K_1$, …, $K_k$) [Algorithm 4].
Return $B$.
```

5.2 Analysis

We first provide two key properties of the batch B of vertices found by Algorithm 3 that are crucial for our final approximation factor and then analyze the running time of the algorithm. Due to space constraints, some proofs are relegated to Appendix C.

Quality of the Schedule. We show that B comprises of vertices whose ancestor sets do not overlap significantly, and further that it is the "maximal" such set.

▶ **Lemma 7.** *The batch B returned by Algorithm 3 satisfies $|\mathcal{A}(B)| > \gamma \sum_{v \in B} |\mathcal{A}(v)|$ and $|\mathcal{E}(B)| > \gamma \sum_{v \in B} |\mathcal{E}(v)|$.*

Proof. Let $B^{(\ell)} \subseteq B$ denote the set containing the first ℓ vertices added to B by the algorithm. We prove the lemma via induction. In the base case, $B^{(1)}$ consists of a single vertex and trivially satisfies the claim. Now suppose that the claim is true for some $\ell \ge 1$ and let v be the $(\ell+1)$-th vertex to be added to B. By Algorithm 3 of Algorithm 3, we add a vertex v into $B^{(\ell)}$ if and only if $|\mathcal{A}(v) \setminus \mathcal{A}(B^{(\ell)})| > \gamma|\mathcal{A}(v)|$ and $|\mathcal{E}(v) \setminus \mathcal{E}(B^{(\ell)})| > \gamma|\mathcal{E}(v)|$. Furthermore, since we

Algorithm 4 Prune$(H, B, K_1, \dots, K_k)$.

```
   Result: New buckets $K_1, \dots, K_k$.
   Input: A graph $H = (V, E)$, a batch $B \subseteq V$, and buckets $K_1, \dots, K_k$.
 1 Obtain estimates $\hat{a}_H(v)$ and $\hat{e}_H(v)$ for all nodes $v \in \cup_{i=1}^{k} K_i$ in the graph $H$.
 2 Let $H' \leftarrow H \setminus \mathcal{A}(B)$
 3 Obtain estimates $\hat{a}_{H'}(v)$ and $\hat{e}_{H'}(v)$ for all nodes $v \in \cup_{i=1}^{k} K_i$ in the graph $H'$.
 4 for $i = k$ to 1 do
 5     for each node $v$ in bucket $K_i$ do
 6         $X \leftarrow \frac{\hat{a}_{H'}(v)}{\hat{a}_H(v)}$.
 7         $Y \leftarrow \frac{\hat{e}_{H'}(v)}{\hat{e}_H(v)}$.
 8         if $X \leq 2\gamma$ or $Y \leq 2\gamma$ then
 9             Remove $v$ from $K_i$.
10 Return the new buckets $K_1, \dots, K_k$.
```

enumerate $\mathcal{A}(v)$ via DFS, our calculation of the cardinality of each of these sets is exact. We now have, $|\mathcal{A}(B^{(\ell+1)})| = |\mathcal{A}(B^{(\ell)})| + |\mathcal{A}(v) \setminus \mathcal{A}(B)| > |\mathcal{A}(B^{(\ell)})| + \gamma|\mathcal{A}(v)|$. By the induction hypothesis, we now have $|\mathcal{A}(B^{(\ell+1)})| > \gamma \sum_{w \in B^{(\ell)}} \mathcal{A}(w) + \gamma\mathcal{A}(v) = \gamma \sum_{w \in B^{(\ell+1)}} \mathcal{A}(w)$. The same proof also holds for $\mathcal{E}(B)$ and the lemma follows. ◀

▶ **Lemma 8.** *If a vertex w was not added to B, it is pruned by Algorithm 4, with high probability. If a vertex v is pruned by Algorithm 4, then $|\mathcal{A}(v) \setminus \mathcal{A}(B)| \leq 4\gamma|\mathcal{A}(v)|$ or $|\mathcal{E}(v) \setminus \mathcal{E}(B)| \leq 4\gamma|\mathcal{E}(v)|$, with high probability.*

Proof. We first prove that any vertex v that is not added to B must be removed from its bucket by Algorithm 4. Any vertex not added to B must have $|\mathcal{A}(v) \setminus \mathcal{A}(B)| \leq \gamma|\mathcal{A}(v)|$. By Lemma 5, $\hat{a}_{H'}(v) \leq 4/3|\mathcal{A}(v) \setminus \mathcal{A}(B)|$ and $\hat{a}_H(v) \geq 2/3|\mathcal{A}(v)|$, with high probability. This must mean that $\frac{\hat{a}_{H'}(v)}{\hat{a}_H(v)} \leq \frac{4/3|\mathcal{A}(v)\setminus\mathcal{A}(B)|}{2/3|\mathcal{A}(v)|} \leq \frac{4/3\gamma|\mathcal{A}(v)|}{2/3|\mathcal{A}(v)|} \leq 2\gamma$. Thus, v will be pruned. The same proof holds for $\hat{e}_{H'}(v)$.

We now prove that the pruning procedure successfully prunes vertices with not too many unique ancestors. In Algorithm 4, by Lemma 5 (setting $\epsilon = 1/3$), we have with high probability, $\hat{a}_{H'}(v) \geq 2/3|\mathcal{A}_{H'}(v)|$. Similarly, with high probability, $\hat{a}_H(v) \leq 4/3|\mathcal{A}_H(v)|$. This means $X = \frac{\hat{a}_{H'}(v)}{\hat{a}_H(v)} \geq \frac{2/3|\mathcal{A}_{H'}(v)|}{4/3|\mathcal{A}_H(v)|} = \frac{1}{2}\left(\frac{|\mathcal{A}_{H'}(v)|}{|\mathcal{A}_H(v)|}\right)$. By the same argument, we also have $Y = \frac{\hat{e}_{H'}(v)}{\hat{e}_H(v)} \geq \frac{1}{2}\left(\frac{|\mathcal{E}_{H'}(v)|}{|\mathcal{E}_H(v)|}\right)$ with high probability.

By Algorithm 4 of Algorithm 4, when we remove a vertex v we have either $X \leq 2\gamma$ or $Y \leq 2\gamma$. By the above, $X, Y \geq \frac{1}{2}(4\gamma) = 2\gamma$. Thus, the largest that $\left(\frac{|\mathcal{A}_{H'}(v)|}{|\mathcal{A}_H(v)|}\right)$ or $\left(\frac{|\mathcal{E}_{H'}(v)|}{|\mathcal{E}_H(v)|}\right)$ can be while still being pruned is 4γ. Thus, with high probability, we have either $\frac{|\mathcal{A}_{H'}(v)|}{|\mathcal{A}_H(v)|} \leq 4\gamma$ or $\frac{|\mathcal{E}_{H'}(v)|}{|\mathcal{E}_H(v)|} \leq 4\gamma$. Since $\mathcal{A}_{H'}(v) = \mathcal{A}(v) \setminus \mathcal{A}(B)$, the claim follows. ◀

The above two lemmas tell us that there are enough unique elements in each batch B, any vertex not added to B will be pruned w.h.p., and the pruning procedure only prunes vertices with a large enough overlap with B w.h.p. This allows us to show the following lemma on the length of the schedule produced by Algorithm 2 for small subgraph H. We first show that we only call Algorithm 3 at most $O\left(\log_{1/\gamma}(\rho)\right)$ times from Algorithm 2 of Algorithm 2.

▶ **Lemma 9.** *The number of batches needed to be scheduled before all vertices in H are scheduled is at most $4\log_{1/4\gamma}(2\rho)$, with high probability.*

Proof. By Lemma 8, each vertex v we do not schedule in a batch B has at least $(1-4\gamma)|\mathcal{A}(v)|$ vertices in $\mathcal{A}(B)$ or at least $(1-4\gamma)|\mathcal{E}(v)|$ edges in $\mathcal{E}(B)$. Since we assumed that all vertices in H have $\leq 2\rho$ ancestors, this means that v can only remain unscheduled for at most $2\log_{1/4\gamma}(4\rho^2)$ batches until $\mathcal{A}(v)$ and $\mathcal{E}(v)$ both become empty (G_v can have at most $4\rho^2$ edges). ◀

Using Lemma 9, we can prove the length of the schedule for H using Algorithm 2. The proof of this lemma is similar to the proof of schedule length of small subgraphs in [19].

▶ **Lemma 10.** *With high probability, the schedule obtained from Algorithm 2 has size at most $\frac{|V_H|}{\gamma M} + 12\rho\log_{1/4\gamma}(2\rho)$ on M processors.*

Proof. By definition of the input, each $\mathcal{A}(v)$ for $v \in V_H$ has at most 2ρ elements. Recall that we schedule all elements in each batch B by duplicating the common shared ancestors such that we obtain a set of independent ancestor sets to schedule. Then, we use a standard list scheduling algorithm to schedule these lists; see Appendix B for a classic list scheduling algorithm. Each vertex in H gets scheduled in exactly one batch since we remove all scheduled vertices from the subgraph used to compute the next batch. Let $B_1, B_2, \dots, B_k$ denote the batches scheduled by Algorithm 2. Let H_i be the subgraph obtained from H by removing batches $B_0, \dots, B_{i-1}$ and adjacent edges. (B_0 is empty.) By Lemma 7, with high probability, for each batch B_i, we have $\sum_{v\in B_i} |\mathcal{A}_{H_i}(v)| \leq \frac{1}{\gamma}|\mathcal{A}_{H_i}(B_i)|$. Let $Z_i = \frac{1}{\gamma}|\mathcal{A}_{H_i}(B_i)|$.

Graham's list scheduling algorithm [13] for independent jobs is known to produce a schedule whose length is at most the total length of jobs divided by the number of machines, plus the length of the longest job. In our case, we treat each ancestor set as one big independent job, and thus for each batch B_i, this bound becomes $Z_i/M + 2\rho$.

Finally Algorithm 2 inserts an idle time of ρ between two successive batches. The total length of the schedule is thus upper bounded by (where k is the number of batches):

$$\begin{aligned}\sum_{i=1}^{k}\left(\frac{Z_i}{M} + 2\rho + \rho\right) &\leq 3\rho \cdot 4\log_{1/4\gamma}(2\rho)\sum_{i=1}^{k}\frac{Z_i}{M} \quad \text{(by Lemma 9)}\\ &\leq 3\rho \cdot 4\log_{1/4\gamma}(2\rho) + \frac{1}{\gamma M}\cdot\sum_{i=1}^{k}|\mathcal{A}_{H_i}(B_i)|\\ &= \frac{|V_H|}{\gamma M} + 12\rho\log_{1/4\gamma}(2\rho)\end{aligned}$$

◀

Running Time. In order to analyze the running time of Algorithm 3, we need a couple of technical lemmas. The key observation is that although computing the ancestor sets $\mathcal{A}(v)$ and $\mathcal{E}(v)$ (in Algorithm 3) of a vertex v takes $O(|\mathcal{E}(v)|)$ time in the worst case, we can bound the total amount of time spent computing these ancestor sets by the size of the ancestor sets scheduled in the batch. There are two main components to the analysis. First, we show that after every iteration of the *pruning* step, the number of vertices in each bucket reduces by at least a constant fraction and hence the sampling procedure is repeated at most $O(\ln n)$ times per batch. Secondly, we use a charging argument to upper bound the amount of time spent enumerating the ancestor sets of sampled vertices.

Finding Stale Vertices. We first argue that with high probability, there are at most $O(\ln n)$ iterations of the while loop in Algorithm 3 of Algorithm 3. Intuitively, in each iteration of the while loop, the number of vertices in any bucket K_i reduces by at least a constant fraction.

▶ **Lemma 11.** *We perform $O(\ln n)$ iterations of sampling and pruning, with high probability, before all buckets are empty. In other words, with high probability, Algorithm 3 of Algorithm 3 runs for $O(\ln n)$ iterations.*

Proof. We prove the lemma for one bucket K_i and by the union bound, the lemma holds for all buckets. First, any sampled vertex which is fresh is added to B. Furthermore, we showed in Lemma 8 that any vertex which is stale is removed from K_i by Algorithm 4. Since the estimates $\hat{a}(v)$ and $\hat{e}(v)$ are within a $\frac{1}{3}$-factor of $|\mathcal{A}(v)|$ and $|\mathcal{E}(v)|$, respectively, we can upper bound $\frac{\hat{a}_{H'}(v)}{\hat{a}_H(v)} \leq 2 \cdot \frac{|\mathcal{A}(v)\setminus\mathcal{A}(B)|}{|\mathcal{A}(v)|}$ (same holds for $\hat{e}(v)$). If a vertex v is stale, then with high probability, we have either $\frac{\hat{a}_{H'}(v)}{\hat{a}_H(v)} \leq \frac{2|\mathcal{A}(v)\setminus\mathcal{A}(B)|}{|\mathcal{A}(v)|} \leq 2\gamma$ or $\frac{\hat{e}_{H'}(v)}{\hat{e}_H(v)} \leq 2\gamma$, and it is removed by Algorithm 4 of Algorithm 4. Since any fresh vertices that are sampled gets added into B and all stale vertices are pruned at the end of each iteration, it only remains to show a large enough number of stale vertices are pruned.

Lemma 17 guarantees that, with high probability, at least $(1-\psi)$-fraction of the vertices in K_i are stale for any constant $\psi \in (0,1)$. Then, Algorithm 4 removes at least $(1-\psi)|K_i|$ vertices in K_i in each iteration. The number of iterations needed is then $\log_{1/(1-\psi)}(|K_i|) = O(\ln n)$.

Since there exists $O(\log \rho)$ buckets and $O(m)$ estimates, we can take the union bound on the probability of success over all buckets and estimates. We obtain, with high probability, $O(\log n)$ iterations are necessary before all buckets are empty. ◀

Charging the Cost of Examining Stale Sets. Here we describe our charging argument that allows us to explictly enumerate the ancestor set of each sampled vertex. Computing the ancestor set of a vertex v takes time $O(|\mathcal{E}(v)|)$ using DFS. Since a fresh vertex gets added to the batch, the cost of computing the ancestor set of a fresh vertex can be easily bounded by the set of edges in $\mathcal{E}(B)$, achieving a total cost, specifically, of $O\left(\frac{1}{\gamma}|\mathcal{E}(B)|\right)$. Our charging argument allows us to bound the cost of computing ancestor sets of sampled stale vertices by charging it to the most recently sampled fresh vertex. Using the above, we provide the runtime of Algorithm 3 below and then the runtime of Algorithm 2.

▶ **Lemma 12.** *Algorithm 3 runs in $O\left(\frac{1}{\gamma}|\mathcal{E}_H(B)|\ln\rho\ln n + |E_H|\ln^3 n\right)$ time, with high probability.*

Proof. The runtime of Algorithm 3 consists of three parts: the time to sample and enumerate ancestor sets, the time to prune stale vertices, and the time to list schedule all vertices in B.

By Lemma 18, the time it takes to enumerate all sampled ancestor sets is $O\left(\frac{1}{\gamma}|\mathcal{E}(B)|\log\rho\log n\right)$ over all iterations of the loops on Algorithm 3 and Algorithm 3 of Algorithm 3.

The time it takes to run Algorithm 4 is $O(|E_H|\ln^2 n)$ since obtaining the estimates for each node (by Lemma 5), creating graph H', and calculating X and Y for each node in the bucket can be done in that time. By Lemma 11, we perform $O(\ln n)$ iterations of pruning, with high probability. Thus, the total time to prune the graph is $O(|E_H|\ln^3 n)$. ◀

▶ **Lemma 13.** *Given a graph $H = (V_H, E_H)$ where $|\mathcal{A}(v)| \leq 2\rho$ for each $v \in V_H$ and parameter $\gamma \in (0, 1/2)$, the time it takes to compute the schedule of H using Algorithm 2 is, with high probability, $O\left(\frac{1}{\gamma}|E_H|\ln\rho\ln n + |E_H|\ln_{1/4\gamma}\rho\ln^3 n + |V_H|\ln M\right)$.*

Proof. By Lemma 9, we perform $O(\log_{1/4\gamma}\rho)$ calls to Algorithm 3. Each call to Algorithm 3 requires $O\left(\frac{1}{\gamma}|\mathcal{E}_H(B)|\ln\rho\ln n + |E_H|\ln^3 n\right)$ time by Lemma 12. However, we know that each vertex (and edges adjacent to it) is scheduled in exactly one batch.

For each batch B, our greedy list scheduling procedure schedules each $\mathcal{A}(v)$ for $v \in B$ greedily and independently by duplicating vertices that appear in more than one ancestor set. Thus, enumerating all the ancestor sets require $O\left(\frac{1}{\gamma}|\mathcal{E}(B)|\right)$ time by Lemma 7. When $M > |B|$, we easily schedule each list on a separate machine in $O\left(\frac{1}{\gamma}|\mathcal{E}(B)|\right)$ time. Otherwise, to schedule the lists, we maintain a priority queue of the machine finishing times. For each list, we greedily assign it to the machine that has the smallest finishing time. We can perform this procedure using $O(M \ln M)$ time. Since $M \leq |B|$, this results in $O(|B| \ln |B|)$ time to assign ancestor sets to machines.

Thus, the total runtime of all calls to Algorithm 3 is

$$\sum_{i=1}^{\log_{1/4\gamma} \rho} O\left(\frac{1}{\gamma}|\mathcal{E}_H(B_i)| \ln \rho \ln n + |E_H| \ln^3 n + \frac{1}{\gamma}|\mathcal{E}(B_i)| + |B| \ln |B|\right)$$
$$= O\left(\frac{1}{\gamma}|E_H| \ln \rho \ln n + |E_H| \ln_{1/4\gamma} \rho \ln^3 n\right).$$

Then, the time it takes to perform Algorithm 2 of Algorithm 2 is $O(1)$ per iteration. Scheduling vertices with no adjacent edges requires $|B| \ln |B| = O(|V_H| \ln M)$ time. Finally, the time it takes to remove each $v \in \mathcal{A}(B)$ and all edges adjacent to v from H for each batch B is $O(|V_H| + |E_H|)$. Doing this for $O(\ln_{1/4\gamma} \rho)$ iterations results in $O(|V_H| + |E_H| \ln_{1/4\gamma} \rho)$ time. ◀

6 Scheduling General Graphs

We now present our main scheduling algorithm for scheduling any DAG $G = (V, E)$ (the full pseudocode is included in Appendix C in our full paper [20]). This algorithm also uses as a subroutine the procedure for estimating the number of ancestors of each vertex in G as described in Section 4. We use the estimates to compute the small subgraphs which we pass into Algorithm 2 to schedule. We produce the small subgraphs by setting the cutoff for the estimates to be $\frac{4}{3}\rho$. This produces small graphs where the number of ancestors of each vertex is upper bounded by 2ρ, with high probability. We present a simplified algorithm below in Algorithm 5. The full pseudocode for our main algorithm is given in Algorithm 10 in our full paper [20].

Algorithm 5 ScheduleGeneralGraph(G).

```
Result: A schedule of the input graph $G = (V, E)$ on $M$ processors.
Input: A directed acyclic task graph $G = (V, E)$.
Let $\mathcal{H} \leftarrow \varnothing$ represent a list of small subgraphs that we will build.
while $G$ is not empty do
    Let $V_H$ be the set of vertices in $G$ where $\hat{a}(v) \leq \frac{4}{3}\rho$ for each $v \in V_H$.
    Compute edge set $E_H$ to be all edges induced by $V_H$.
    Add $H = (V_H, E_H)$ to $\mathcal{H}$.
    Remove $V_H$ and all incident edges from $G$.
end
for $H \in \mathcal{H}$ in the order they were added do
    Call ScheduleSmallSubgraph($H$) to obtain a schedule of $H$. [Algorithm 2]
end
```

Quality of the Schedule and Running Time. Let OPT be the length of the optimal schedule. We first give two bounds on OPT, and then relate them to the length of the schedule found by our algorithm. A detailed set of proofs is provided in Appendix C.2.

Our main algorithm, Algorithm 5, partitions the vertices of G into small subgraphs $H \in \mathcal{H}$. It does so based on estimates of ancestor set sizes. We first lower bound OPT by working with exact ancestor set sizes. Since the schedule produced by our algorithm cannot have length smaller than OPT, this process also provides a lower bound on our schedule length. Then, we show that Algorithm 5 does not output more small subgraphs than the number of subgraphs produced by working with exact ancestor set sizes, with high probability.

The crucial fact in obtaining our final runtime is that producing the estimates of the number of ancestors of each vertex requires $\tilde{O}(|V| + |E|)$ time *in total* over the course of finding all small subgraphs. Together, these facts allow us to obtain Theorem 14.

▶ **Theorem 14.** *On input graph $G = (V, E)$, Algorithm 5 produces a schedule of length at most $O\left(\frac{\ln \rho}{\ln \ln \rho} \cdot (\mathsf{OPT} + \rho)\right)$ and runs in time $O\left(n \ln M + \frac{m \ln^3 n \ln \rho}{\ln \ln \rho}\right)$, with high probability.*

Proof. By Lemma 21, Algorithm 3 is called at most L times. Then, since each vertex is in at most one small subgraph (and hence each edge is in at most one small subgraph), the total runtime for all the calls (by Lemma 13) is

$$\begin{aligned}&\sum_{i=1}^{L} O\left(\frac{1}{\gamma}|E_{H_i}| \ln \rho \ln n + |E_{H_i}| \ln_{1/4\gamma} \rho \ln^3 n + |V_{H_i}| \ln M\right)\\ &= O\left(\frac{1}{\gamma}|E| \ln \rho \ln n + |E| \ln_{1/4\gamma} \rho \ln^3 n + |V| \ln M\right).\end{aligned}$$

Furthermore, each iteration of Algorithm 5 requires estimating $\hat{a}(v)$ for a set of vertices v, adding v to H, and checking all successors of v. First, we show that $\hat{a}(v)$ is computed at most twice for each vertex in V, and then, we show that the rest of the steps are efficient.

Each vertex contained in the queue, Q, in Algorithm 3, either does not have any ancestors, or all of its ancestors are in H (the current subgraph). If a vertex $v \in Q$ was not added to H during iteration i, then it must have at least one ancestor in iteration i and no ancestors in iteration $i + 1$. Since v has no ancestors in iteration $i + 1$, it must be added to H_{i+1}. The time it takes to compute the estimate for one vertex is $O(\ln^2 n)$. Thus, the total time it takes to compute the estimate of the number of ancestors of all vertices is $O\left(m \ln^2 n\right)$. Adding v to H and checking all successors can be done in $O(m)$ time in total across all vertices and subgraphs. Finally removing each $v \in H$ from G can be done in $O(m)$ time in total for all v.

As earlier, we use $\gamma = \sqrt{\ln \rho}$, so the total runtime summing the above can be upper bounded by $O\left(n \ln M + \frac{m \ln^3 n \ln \rho}{\ln \ln \rho}\right)$. Thus, the algorithm produces a schedule of length $O\left(\frac{\ln \rho}{\ln \ln \rho} \cdot (\mathsf{OPT} + \rho)\right)$ (by Theorem 22) and the total runtime of the algorithm is $O\left(n \ln M + \frac{m \ln^3 n \ln \rho}{\ln \ln \rho}\right)$, with high probability. ◀

Theorem 14 gives the main result of our paper stated informally in Theorem 1 of the introduction.

References

1 Ishfaq Ahmad and Yu-Kwong Kwok. On exploiting task duplication in parallel program scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):872–892, September 1998. doi:10.1109/71.722221.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 20–29, 1996. doi:10.1145/237814.237823.

3 Evripidis Bampis, Aristotelis Giannakos, and Jean-Claude König. On the complexity of scheduling with large communication delays. *European Journal of Operational Research*, 94:252–260, 1996.

4 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, RANDOM '02, pages 1–10, 2002.

5 Doruk Bozdag, Fusun Ozguner, and Umit V Catalyurek. Compaction of schedules and a two-stage approach for duplication-based DAG scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 20(6):857–871, 2008.

6 Peter Brucker. *Scheduling Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2010.

7 Israel Casas, Javid Taheri, Rajiv Ranjan, Lizhe Wang, and Albert Y Zomaya. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. *Future Generation Computer Systems*, 74:168–178, 2017.

8 P. Chassaing and L. Gerin. Efficient estimation of the cardinality of large data sets. *Discrete Mathematics & Theoretical Computer Science*, pages 419–422, 2006.

9 S. Darbha and D. P. Agrawal. Optimal scheduling algorithm for distributed-memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 9:87–95, 1998.

10 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via LP hierarchies and clustering. In *FOCS*, 2020.

11 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via LP hierarchies and clustering II: weighted completion times on related machines. In *SODA 2021*, pages 2958–2977. SIAM, 2021. doi:10.1137/1.9781611976465.176.

12 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In *AofA: Analysis of Algorithms*, pages 137–156, 2007. URL: https://hal.inria.fr/hal-00406166.

13 R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17:416–429, 1969.

14 R. L. Graham. Bounds on multiprocessing anomalies and related packing algorithms. In *Proceedings of the May 16-18, 1972, Spring Joint Computer Conference*, AFIPS '72 (Spring), pages 205–217, New York, NY, USA, 1971. Association for Computing Machinery. doi:10.1145/1478873.1478901.

15 Claire Hanen and Alix Munier. An approximation algorithm for scheduling dependent tasks on m processors with small communication delays. *Discret. Appl. Math.*, 108(3):239–257, 2001. doi:10.1016/S0166-218X(00)00179-7.

16 J.A. Hoogeveen, J.K. Lenstra, and B. Veltman. Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, 16(3):129–137, 1994. doi:10.1016/0167-6377(94)90024-8.

17 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 41–52, 2010. doi:10.1145/1807085.1807094.

18 Janardhan Kulkarni, Shi Li, Jakub Tarnawski, and Minwei Ye. Hierarchy-based algorithms for minimizing makespan under precedence and communication constraints. In *Proceedings of the Fortieth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2020.

19 Renaud Lepère and Christophe Rapine. An asymptotic $O(\ln \rho / \ln \ln \rho)$-approximation algorithm for the scheduling problem with duplication on large communication delay graphs. In *STACS*, volume 2285 of *Lecture Notes in Computer Science*, pages 154–165, 2002. doi:10.1007/3-540-45841-7_12.

20 Quanquan C. Liu, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang. Scheduling with communication delay in near-linear time. *CoRR*, abs/2108.02770, 2021. arXiv:2108.02770.

21 Biswaroop Maiti, Rajmohan Rajaraman, David Stalfa, Zoya Svitkina, and Aravindan Vijayaraghavan. Scheduling precedence-constrained jobs on related machines withcommunication delay. In *FOCS*, 2020.

22 Alix Munier. Approximation algorithms for scheduling trees with general communication delays. *Parallel Computing*, 25(1):41–48, 1999.

23 Alix Munier and Claire Hanen. Using duplication for scheduling unitary tasks on m processors with unit communication delays. *Theoretical Computer Science*, 178(1):119–127, 1997. doi:10.1016/S0304-3975(97)88194-7.

24 Alix Munier and Jean-Claude König. A heuristic for a scheduling problem with communi-cation delays. *Operations Research*, 45(1):145–147, 1997.

25 Rasmus Pagh, Morten Stöckel, and David P. Woodruff. Is min-wise hashing optimal for summarizing set intersection? In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '14, pages 109–120, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2594538.2594554.

26 Michael A. Palis, Jing-Chiou Liou, and David S. L. Wei. Task clustering and scheduling for distributed memory parallel architectures. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):46–55, 1996.

27 Christos H. Papadimitriou and Mihalis Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM journal on computing*, 19(2):322–328, 1990.

28 Christophe Picouleau. New complexity results on scheduling with small communication delays. *Discret. Appl. Math.*, 60(1-3):331–342, 1995. doi:10.1016/0166-218X(94)00063-J.

29 Victor J Rayward-Smith. UET scheduling with unit interprocessor communication delays. *Discrete Applied Mathematics*, 18(1):55–71, 1987.

30 Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Syst.*, 15(2):208–229, June 1990. doi:10.1145/78922.78925.

A Count-Distinct Estimator [4]

We provide the algorithm of Bar-Yossef et al. [4] in Algorithm 6 (Appendix A) in our full paper [20]. The algorithm of [4] works as follows. Provided a multiset S of elements where $n = |S|$, we pick $t = \frac{c}{\varepsilon^2}$ where c is some fixed constant $c \geq 1$ and $\mathcal{H}$, a 2-univesal hash family. Then, we choose $O(\log n)$ hash functions from $\mathcal{H}$ uniformly at random, without replacement. For each hash function $h_i : [n] \to [n^3]$ $(i = O(\log n))$, we maintain a balanced binary tree T_i of the *smallest* t values seen so far from the hash outputs of h_i. Initially, all T_i are empty. We iterate through S and for each $a_j \in S$, we compute $h_i(a_j)$ using each h_i that we picked; we update T_i if $h_i(a_j)$ is smaller than the largest element in T_i or if the size of T_i is smaller than t. After iterating through all of S, for each T_i, we add the largest value of each tree T_i to a list L. Then, we sort L and find the median value ℓ (using the *median trick*). We return tn^3/ℓ as our estimate.

We now show how to use Bar-Yossef et al. [4] to get our desired mergeable estimator. Let $\mathcal{T}_X$ be the set of trees $T_i \in \mathcal{T}_X$ maintained for the estimator defined by Bar-Yossef et al. [4] for multiset X. Since each T_i has size at most $O(t) = O\left(\frac{1}{\varepsilon^2}\right)$, the total space required to store all T_i is $O\left(\frac{1}{\varepsilon^2}\log^2 n\right)$ in bits. We can initialize our estimator on input d by picking a set of random hash functions: $h_1, \ldots, h_{d\log n} \in \mathcal{H}$. Let H be the set of picked hash function. Then, for each set S, we initialize $d \log n$ trees $T_i \in \mathcal{T}_S$ and maintain $\mathcal{T}_S$ in memory. The elements of T_i are computed using $h_i \in H$. Let $\mathcal{D}_S$ denote the estimator for S. Using $\mathcal{T}_S$ for set S, we can implement the following functions (pseudocode for the three functions can be found in Algorithm 6):

- **CountDistinctEstimator.insert($\mathcal{D}_S, x$)**: Insert $h_i(x)$ into $T_i \in \mathcal{T}_S$ for each $i \in [d \log n]$. If T_i has size greater than t, delete the largest element of T_i.
- **CountDistinctEstimator.merge($\mathcal{D}_{S_1}$, $\mathcal{D}_{S_2}$)**: Here we assume that the same set of hash functions are used for both $\mathcal{D}_{S_1}$ and $\mathcal{D}_{S_2}$. For each pair of $T_{1,i} \in \mathcal{T}_{S_1}$ and $T_{2,i} \in \mathcal{T}_{S_2}$ for hash function h_i, build a new tree T_i by taking the t smallest elements from $T_{1,i} \cup T_{2,i}$.
- **CountDistinctEstimator.estimateCardinality($\mathcal{D}_S$)**: Let ℓ be the median value of the largest values of the trees $T_i \in \mathcal{T}_S$. Return tN/ℓ.

Algorithm 6 Initialize New CountDistinctEstimator.

```
Input: $\mathcal{D}_S, \mathcal{T}_S, t, h_i \in \mathcal{H}$ are as defined above for multiset $S$. Let $n = |S|$.
CountDistinctEstimator.insert($\mathcal{D}_S, x$):
    for $T_i \in \mathcal{T}_S$ do
        Compute $h_i(x)$.
        if $T_i$ has less than $t$ elements or $h_i(x)$ is smaller than the largest value in $T_i$ then
            Insert $h_i(x)$ into $T_i$.
        end
        if $T_i$ has more than $t$ elements then
            Remove the largest element in $T_i$.
        end
    end
CountDistinctEstimator.merge($\mathcal{D}_{S_1}$, $\mathcal{D}_{S_2}$):
    for $T_{1,i} \in \mathcal{T}_{S_1}$, $T_{2,i} \in \mathcal{T}_{S_2}$ do
        Perform inorder traversal of $T_{1,i}$ and $T_{2,i}$ to obtain non-decreasing lists of elements, $L_{1,i}$ and $L_{2,i}$.
        Merge $L_{1,i}$ and $L_{2,i}$ to obtain a new non-decreasing list of elements, $L$.
        Build a new balanced binary tree from the first $t$ elements of $L$.
    end
CountDistinctEstimator.estimateCardinality($\mathcal{D}_S$):
    for $T_i \in \mathcal{T}_S$ do
        Insert largest element of $T_i$ into list $L$.
    end
    Sort $L$.
    Let $\ell$ be the median of $L$.
    Return $tn^3/\ell$.
```

The estimator provided in Bar-Yossef et al. [4] satisfies the following lemmas as proven in [8] (specifically it is proven that the estimator is unbiased):

▶ **Lemma 15** ([4]). *The Bar-Yossef et al. [4] estimator is an $\left(\varepsilon, \frac{1}{n^d}, O\left(\frac{1}{\varepsilon^2}\log^2 n\right)\right)$-estimator for the count-distinct problem.*

▶ **Lemma 16.** *Furthermore, the insert, merge, and estimate cardinality functions of the Bar-Yossef et al. [4] estimator can be implemented in $O\left(\frac{1}{\varepsilon^2}\log^2 n\right)$ time.*

Proof. **CountDistinctEstimator.insert** requires $O(\log t)$ time to insert $h_i(x)$ and $O(\log t)$ time to remove the largest element. Thus, this method requires $O\left(\log\left(\frac{1}{\varepsilon}\right)\right) = O\left(\frac{1}{\varepsilon^2}\right)$ time. **CountDistinctEstimator.merge** requires $O(t) = O\left(\frac{1}{\varepsilon^2}\right)$ time to merge $T_{1,i}$ and $T_{2,i}$ and also $O(t)$ time to build the new tree. Finally, **CountDistinctEstimator.estimateCardinality** requires $O(\log n)$ time to create the list L and $O(\log n \log\log n)$ time to sort and find the median. ◀

Estimating the Number of Ancestors and Edges. Using the count-distinct estimator described above, we can provide our full algorithms for estimating the number of ancestors and the number of edges in the induced subgraph of every vertex in a given input graph.

Our complete algorithm for estimating the number of ancestors $\hat{a}(v)$ of every vertex in an input graph is given in Algorithm 7. Our algorithm for finding $\hat{e}(v)$ for every vertex in the input graph is given in Algorithm 9 in our full paper [20].

Algorithm 7 Estimate Number of Ancestors.

```
Result: Estimate $\hat{a}_H(v)$ such that $(1-\epsilon)|\mathcal{A}_H(v)| \leq \hat{a}_H(v) \leq (1+\epsilon)|\mathcal{A}_H(v)|, \forall v \in V$.
Input: A graph $H = (V, E)$.
Topologically sort all the vertices in $H$.
for vertex $w$ in the topological order of vertices in $H$ do
    Let $\mathsf{Pred}(w)$ be the set of predecessors of $w$.
    Let $\mathcal{D}_w \leftarrow$ New $\left(\varepsilon, \frac{1}{n^d}, O\left(\frac{1}{\varepsilon^2}\log^2 n\right)\right)$-CountDistinctEstimator for $w$.
    CountDistinctEstimator.insert$(\mathcal{D}_w, w)$
    for $v \in \mathsf{Pred}(w)$ do
        $\mathcal{D}_w =$ CountDistinctEstimator.merge$(\mathcal{D}_w, \mathcal{D}_v)$.
    end
    $\hat{a}_H(w) =$ CountDistinctEstimator.estimateCardinality$(\mathcal{D}_w)$
end
```

B List Scheduling

Here we provide a brief description of the classic Graham list scheduling algorithm [14]. For our purposes, we are given a set of vertices and their ancestors. We duplicate the ancestors for each vertex v so that each vertex and its ancestors is scheduled as a *single* unit with job size equal to the *number of ancestors* of v. Then, we perform the following greedy procedure: for each vertex v, we sequentially assign v to the machine $M_i \in \mathcal{M}$ with *smallest load* (i.e. load is defined by the jobs lengths of all jobs assigned to it). We can maintain loads of the machines in a heap to determine the machine with the lowest load at any time. To schedule n jobs using this procedure requires $O(n \ln M)$ time.

C Deferred Proofs

In this section, we include all of the proofs deferred from the main text.

C.1 Runtime of Scheduling Small Subgraphs

▶ **Lemma 17.** *For any constant $d \geq 1$ and $\psi > 0$, there is a constant $c \geq 1$ such that, with probability at least $1 - \frac{1}{n^d}$, at most a ψ-fraction of remaining nodes in each bucket are fresh after sampling $c \ln n$ stale vertices consecutively.*

Proof. The main approach behind the proof is that we show that for any bucket where a constant fraction ψ of the vertices in the bucket are fresh, for any $c \ln n$ consecutively sampled vertices, we expect to see $\psi c \ln n$ fresh vertices. Furthermore, we show a concentration bound around this expected number of vertices using the Chernoff bound. Thus, we can conclude that if we see $c \ln n$ stale vertices consecutively (with no good vertices), then with high probability, at most a small constant fraction of the remaining nodes in the bucket is fresh.

Algorithm 3 samples the vertices in each bucket K_i consecutively, uniformly at random without replacement, until a new fresh vertex is found or at least $c \ln n$ stale vertices are sampled consecutively. Let F be the set of fresh and stale vertices sampled (and removed) so far from bucket K_i *up to the most recent time* a fresh vertex was sampled from K_i (i.e. F includes all vertices sampled including and up to the most recent fresh vertex sampled from K_i). Let ψ be some fraction $0 < \psi < 1$. Suppose at most a ψ-fraction of the vertices in bucket K_i are fresh after removing the previously sampled F vertices. Such an ψ exists for every bucket with at least one fresh vertex and one stale vertex after doing such removals. (In the case when all vertices in the bucket are fresh, all vertices from that bucket will be sampled and added to B. If all vertices in the bucket are stale, then $c \ln n$ stale vertices will be sampled immediately.)

From here on out, we assume the bucket K_i only contains the remaining vertices after the previously sampled F vertices were removed. We assume the number of remaining vertices in K_i is more than $c \ln n$. The probability that each of the next sampled vertices is a fresh vertex is at least ψ. The expected number of fresh vertices in the $c \ln n$ samples from K_i is lower bounded by:

$$\sum_{i=1}^{c \ln n} \left(i \cdot \binom{c \ln n}{i} \psi^i (1-\psi)^{c \ln n - i} \right) = \psi c \ln n.$$

Suppose for our analysis that we only remove stale vertices when we sample them (and not fresh vertices). The above is a lower bound, in this setting, on the expected number of sampled fresh vertices from K_i since ψ is the fraction of fresh vertices after removing F; if we remove more stale vertices, the fraction of fresh vertices cannot decrease so ψ upper bounds the fraction of fresh vertices in K_i as we remove more stale vertices. This assumption is the same as our algorithm when all $c \ln n$ sampled vertices are stale.

By the Chernoff bound, the probability that we sample less than $(1-\varepsilon)\psi c \ln n$ fresh vertices is less than $\exp\left(-\frac{\varepsilon^2 \psi c \ln n}{2}\right)$. When $c > \frac{1}{(1-\varepsilon)\psi}$, $(1-\varepsilon)\psi c \ln n \geq 1$ for any $0 < \varepsilon < 1$ and $0 < \psi < 1$. Then, the probability that no fresh vertices are sampled is less than $\exp\left(-\frac{\varepsilon^2 \psi c \ln n}{2}\right) = n^{\frac{-\varepsilon^2 \psi c}{2}}$. It is easy to consider the case for constant $\psi \in (0,1)$. If $\psi = o(1)$, then there exists a constant ϕ for which at most a ϕ-fraction of the vertices in K_i are fresh. If $\psi = \omega(1)$, then the probability becomes super-polynomially small. We can sample $c \ln n$ vertices for large enough constant $c \geq \frac{6d}{\varepsilon^2 \psi}$ such that with probability at least $1 - \frac{1}{n^d}$ for any constant $d \geq 1$, there exists less than ψ-fraction of vertices in the bucket that are fresh if the next $c \ln n$ sampled vertices are stale. The factor of 6 in the bound $c \geq \frac{6d}{\varepsilon^2 \psi}$ is useful when we take the union bound over multiple trials (at most $O(n^2)$) for all buckets used during the course of this algorithm. ◀

▶ **Lemma 18.** *With high probability, the total runtime of enumerating the ancestor sets of all sampled vertices in Algorithm 3 is $O\left(\frac{1}{\gamma}|\mathcal{E}(B)| \ln \rho \ln n\right)$. In other words, the total runtime of performing all iterations of Algorithm 3 of Algorithm 3 is $O\left(\frac{1}{\gamma}|\mathcal{E}(B)| \ln \rho \ln n\right)$, with high probability.*

Proof. First, we calculate the runtime of enumerating the ancestor sets of each element of B. By Lemma 7, $|\mathcal{E}(B)| \geq \gamma \sum_{v \in B} |\mathcal{E}(v)|$. Hence, the amount of time to enumerate all ancestor sets of every vertex in B is at most $\frac{1}{\gamma}|\mathcal{E}(B)|$.

We employ the following charging scheme to calculate the total time necessary to enumerate the ancestor sets of all sampled stale vertices. Let u be the most recent vertex added to B from some bucket K_i. We charge the cost of enumerating the ancestor sets of all stale vertices sampled after u to the cost of enumerating the ancestor set of u. Since we sample at most $O(\log n)$ consecutive stale vertices from each bucket before moving to the next bucket, u gets charged with at most the work of enumerating $O(\log \rho \log n)$ vertices from the same or smaller buckets. With high probability, the largest ancestor set in bucket K_i has a size at most four times the smallest ancestor set size. Since we sample vertices in decreasing bucket size, we charge at most $O(|\mathcal{E}(v)| \log \rho \log n)$ work to v.

By our bound on the cost of enumerating all ancestor sets of vertices in B, the additional charged cost results in a total cost of $\sum_{v \in B} |\mathcal{E}(v)| \cdot O(\log \rho \log n) = O(\frac{1}{\gamma} |\mathcal{E}(B)| \log \rho \log n)$. ◀

C.2 Quality of the Schedule Produced by the Main Algorithm

Assuming we are working with exact ancestor set sizes, we would wind up with vertex sets $V_1 \triangleq \{v \in V : |\mathcal{A}(v)| \le \rho\}$ and, inductively, for $i > 1$, $V_i \triangleq \{v \in V \setminus \bigcup_{j=1}^{i-1} V_j : |\mathcal{A}(v) \setminus \bigcup_{j=1}^{i-1} V_j| \le \rho\}$. Let L be the maximum index such that V_L is nonempty.

The following lemma follows a similar argument as that found in Lepere-Rapine [19] (although we have simplified the analysis). We repeat it here for completeness.

▶ **Lemma 19.** $\mathsf{OPT} \ge (L-1)\rho$.

Proof. We show by induction on i that in any valid schedule, there exists a job $v \in V_i$ that cannot start earlier than time $(i-1)\rho$. Given that, the job in V_L starts at time at least $(L-1)\rho$ in OPT, proving the lemma.

The base case of $i = 1$ is trivial. For the induction step, consider a job $v \in V_{i+1}$. This job has at least ρ ancestors in V_i (call this set $A = \mathcal{A}(v) \cap V_i$), since if it had less, v would be in V_i itself. All jobs in A start no earlier than $(i-1)\rho$ by the induction hypothesis. There are two cases. If all of the jobs in A are executed on the same machine as v, then it would take at least ρ units of time for them to finish before v can start. If at least one job in A is executed on a different machine than v, then it would take ρ units of time to communicate the result. In either case, v would start later than the first job in A by at least ρ, and thus no earlier than $i \cdot \rho$. ◀

▶ **Lemma 20.** $\mathsf{OPT} \ge |V|/M$.

Proof. Every job has to be scheduled on at least one machine, and the makespan is at least the average load on any machine. ◀

We show that our general algorithm only calls the schedule small subgraph procedure at most L times, w.h.p.

▶ **Lemma 21.** *With high probability, Algorithm 5 calls Algorithm 2 at most L times on input graph $G = (V, E)$.*

Proof. By construction, the V_i's are inductively defined by stripping all vertices with ancestor sets at most ρ in size. With high probability, our estimates $\hat{a}(v)$ are at most $\frac{4}{3}|\mathcal{A}(v)|$. Algorithm 5 only takes vertex v into the subgraph H if $\hat{a}(v) \le \frac{4}{3}\rho$. By Lemma 5, $\frac{2}{3}|\mathcal{A}(v)| \le \hat{a}(v) \le \frac{4}{3}|\mathcal{A}(v)|$. Then, $|\mathcal{A}(v)| \le \frac{3}{2}\hat{a}(v) \le \frac{3}{2} \cdot \frac{4}{3}\rho = 2\rho$. Furthermore, since $|\mathcal{A}(v)| \ge \frac{3}{4} \cdot \hat{a}(v)$, if $\hat{a}(v) = \frac{4}{3}\rho$, then $|\mathcal{A}(v)| \ge \rho$. Hence, all vertices with height ρ are added into H, with high probability. Taken together, this means that all vertices of V_i (even if their ancestor sets are maximally overestimated) are contained in the small graphs H produced by iterations one through i of Algorithm 5 of Algorithm 5 Since V_L was chosen to be the last non-empty set, we know our algorithm runs for at most L iterations, with high probability. ◀

▶ **Theorem 22.** *Algorithm 5 produces a schedule of length at most* $O\left(\frac{\ln \rho}{\ln \ln \rho}\right) \cdot (\mathsf{OPT} + \rho)$.

Proof. In Algorithm 2, by Lemma 10, the schedule length obtained from any small subgraph H is $\frac{|V_H|}{\gamma M} + 12\rho \log_{1/4\gamma}(2\rho)$.

Let $\mathcal{H}$ be the set of all small subgraphs Algorithm 5 sends to Algorithm 2 to be scheduled. By Lemma 21, there are at most L of them. Each vertex trivially appears in at most one subgraph. Then the total length of our schedule is given by

$$\begin{aligned}\sum_{H \in \mathcal{H}} \left(\frac{|V_H|}{\gamma M} + 12\rho \log_{1/4\gamma}(2\rho)\right) &= \sum_{H \in \mathcal{H}} \frac{|V_H|}{\gamma M} + \sum_{H \in \mathcal{H}} 12\rho \log_{1/4\gamma}(2\rho) \\ &\leq \frac{|V|}{\gamma M} + L\left(12\rho \log_{1/4\gamma}(2\rho)\right).\end{aligned}$$

By Lemmas 19 and 20, this last quantity is upper bounded by

$$OPT \cdot \left(\frac{1}{\gamma} + 12\log_{1/4\gamma}(2\rho)\right) + \rho \cdot 12\log_{1/4\gamma}(2\rho)$$

Setting $\gamma = 1/\sqrt{\ln \rho}$ gives our bound of $(OPT + \rho) \cdot O(\frac{\ln \rho}{\ln \ln \rho})$. ◀

Extending the Reach of the Point-To-Set Principle

Jack H. Lutz
Department of Computer Science, Iowa State University, Ames, IA, USA

Neil Lutz
Computer Science Department, Swarthmore College, PA, USA

Elvira Mayordomo
Departamento de Informática e Ingeniería de Sistemas,
Instituto de Investigación en Ingeniería de Aragón, University of Zaragoza, Spain

Abstract

The *point-to-set principle* of J. Lutz and N. Lutz (2018) has recently enabled the theory of computing to be used to answer open questions about fractal geometry in Euclidean spaces $\mathbb{R}^n$. These are classical questions, meaning that their statements do not involve computation or related aspects of logic.

In this paper we extend the *reach* of the point-to-set principle from Euclidean spaces to arbitrary separable metric spaces X. We first extend two fractal dimensions – computability-theoretic versions of classical Hausdorff and packing dimensions that assign dimensions $\dim(x)$ and $\mathrm{Dim}(x)$ to *individual points* $x \in X$ – to arbitrary separable metric spaces and to arbitrary gauge families. Our first two main results then extend the point-to-set principle to arbitrary separable metric spaces and to a large class of gauge families.

We demonstrate the power of our extended point-to-set principle by using it to prove new theorems about classical fractal dimensions in hyperspaces. (For a concrete computational example, the stages $E_0, E_1, E_2, \ldots$ used to construct a self-similar fractal E in the plane are elements of the hyperspace of the plane, and they converge to E in the hyperspace.) Our third main result, proven via our extended point-to-set principle, states that, under a wide variety of gauge families, the classical packing dimension agrees with the classical upper Minkowski dimension on *all* hyperspaces of compact sets. We use this theorem to give, for all sets E that are analytic, i.e., $\mathbf{\Sigma}^1_1$, a tight bound on the packing dimension of the hyperspace of E in terms of the packing dimension of E itself.

2012 ACM Subject Classification Theory of computation → Computability

Keywords and phrases algorithmic dimensions, geometric measure theory, hyperspace, point-to-set principle

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.48

Related Version *Extended Version*: `https://arxiv.org/abs/2004.07798`

Funding *Jack H. Lutz*: Research supported in part by National Science Foundation grants 1545028 and 1900716. Part of this work was done during a visit to the Institute for Mathematical Sciences at the National University of Singapore.
Neil Lutz: Part of this work was done while this author was at the University of Pennsylvania and at Iowa State University.
Elvira Mayordomo: Research supported in part by projects TIN2016-80347-R and PID2019-104358RB-I00 granted by MCIN/AEI and Government of Aragon COSMOS Group T64_20R, and partly done during a visit to the Institute for Mathematical Sciences at the National University of Singapore.

Acknowledgements We thank anonymous reviewers for several observations that have improved this paper.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 48; pp. 48:1–48:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

It is rare for the theory of computing to be used to answer open mathematical questions – especially questions in continuous mathematics – whose statements do not involve computation or related aspects of logic.[1] The *point-to-set principle* [22], described below, has enabled several recent developments that do exactly this. This principle has been used to obtain strengthened lower bounds on the Hausdorff dimensions of generalized Furstenberg sets [27], extend the fractal intersection formula for Hausdorff dimension from Borel sets to arbitrary sets [25], and prove that Marstrand's projection theorem for Hausdorff dimension holds for any set E whose Hausdorff and packing dimensions coincide, whether or not E is analytic [26].[2] (See [5, 6, 23, 24] for reviews of these developments.) More recently, the point-to-set principle has been used to prove that $V = L$ implies that the maximal thin co-analytic set has Hausdorff dimension 1 [40] and that the Continuum Hypothesis implies that every $s \in (0, 1]$ is the Hausdorff dimension of a Hamel basis of the vector space $\mathbb{R}$ over the field $\mathbb{Q}$ [21]. These applications of the point-to-set principle all concern fractal geometry in Euclidean spaces $\mathbb{R}^n$.[3]

This paper extends the reach of the point-to-set principle beyond Euclidean spaces. To explain this, we first review the point-to-set principle to date. (All quantities defined in this intuitive discussion are defined precisely later in the paper.) The two best-behaved classical fractal dimensions, Hausdorff dimension and packing dimension, assign to every subset E of a Euclidean space $\mathbb{R}^n$ dimensions $\dim_{\mathrm{H}}(E)$ and $\dim_{\mathrm{P}}(E)$, respectively. When E is a "smooth" set that intuitively has some integral dimension between 0 and n, the Hausdorff and packing dimensions agree with this intuition, but more complex sets E may have any real-valued dimensions satisfying $0 \leq \dim_{\mathrm{H}}(E) \leq \dim_{\mathrm{P}}(E) \leq n$. Hausdorff and packing dimensions have many applications in information theory, dynamical systems, and other areas of science [2, 7, 14, 35].

Early in this century, algorithmic versions of Hausdorff and packing dimensions were developed to quantify the information densities of various types of data. The computational resources allotted to these algorithmic dimensions range from finite-state to computable enumerability and beyond, but the point-to-set principle concerns the computably enumerable algorithmic dimensions introduced in [20, 1].[4] These assign to each *individual point* x in a Euclidean space $\mathbb{R}^n$ an *algorithmic dimension* $\dim(x)$ and a *strong algorithmic dimension* $\mathrm{Dim}(x)$. The point-to-set principle of [22] is a complete characterization of the classical Hausdorff and packing dimensions in terms of oracle relativizations of these very non-classical dimensions of individual points. Specifically, the point-to-set principle says that, for every set E in a Euclidean space $\mathbb{R}^n$,

$$\dim_{\mathrm{H}}(E) = \min_{A \subseteq \mathbb{N}} \sup_{x \in E} \dim^A(x) \tag{1.1}$$

[1] We use the adjective "classical" for theorems and questions whose statements do not involve computability or logic, regardless of when they were proven or formulated. A "classical" theorem can thus be very new.

[2] These very non-classical proofs of new classical theorems have provoked new work in the fractal geometry community. Orponen [34] has very recently used a discretized potential-theoretic method of Kaufman [16] and tools of Katz and Tao [15] to give a new, classical proof of the two main theorems of [26].

[3] Applications of the theory of computing – specifically Kolmogorov complexity – to discrete mathematics are more numerous and are surveyed in [19]. Other applications to continuous mathematics, not involving the point-to-set principle, include theorems in descriptive set theory [32, 12, 17], Riemannian moduli space [43], and Banach spaces [18].

[4] These have also been called "constructive" dimensions and "effective" dimensions by various authors.

and

$$\dim_{\mathrm{P}}(E) = \min_{A\subseteq\mathbb{N}} \sup_{x\in E} \mathrm{Dim}^A(x), \tag{1.2}$$

where the dimensions on the right are relative to the oracle A. The point-to-set principle is so named because it enables one to use a lower bound on the relativized algorithmic dimension of a single, judiciously chosen *point* in a set E to prove a lower bound on the classical dimension of the *set* E.

The classical Hausdorff and packing dimensions work not only in Euclidean spaces, but in arbitrary metric spaces. In contrast, nearly all work on algorithmic dimensions to date (the exception being [29]) has been in Euclidean spaces or in spaces of infinite sequences over finite alphabets. Our objective here is to significantly reduce this gap by extending the theory of algorithmic dimensions, along with the point-to-set principle, to arbitrary separable metric spaces. (A metric space X is *separable* if it has a countable subset D that is *dense* in the sense that every point in X has points in D arbitrarily close to it.)

In parallel with extending algorithmic dimensions to separable metric spaces, we also extend them to arbitrary gauge families. It was already explicit in Hausdorff's original paper [8] that his dimension could be defined via various "lenses" that we now call *gauge functions.* In fact, one often uses, as we do here, a *gauge family* φ, which is a one-parameter family of gauge functions φ_s for $s \in (0,\infty)$. For each separable metric space X, each gauge family φ, and each set $E \subseteq X$, the classical *φ-gauged Hausdorff dimension* $\dim^{\varphi}_{\mathrm{H}}(E)$ and *φ-gauged packing dimension* $\dim^{\varphi}_{\mathrm{P}}(E)$ are thus well-defined. In this paper, for each separable metric space X, each gauge family φ, and each point $x \in X$, we define the *φ-gauged algorithmic dimension* $\dim^{\varphi}(x)$ and the *φ-gauged strong algorithmic dimension* $\mathrm{Dim}^{\varphi}(x)$ of the point x. We should mention here that there is a particular gauge family θ that gives the "un-gauged" dimensions in the sense that the identities $\dim^{\theta}_{\mathrm{H}}(E) = \dim_{\mathrm{H}}(E)$, $\dim^{\theta}_{\mathrm{P}}(E) = \dim_{\mathrm{P}}(E)$, $\dim^{\theta}(x) = \dim(x)$, and $\mathrm{Dim}^{\theta}(x) = \mathrm{Dim}(x)$ always hold.

Our first two main results (Theorems 4.1 and 4.2) extend the point-to-set principle to arbitrary separable metric spaces and a wide variety of gauge families, proving that, for every separable metric space X, every gauge family φ satisfying mild asymptotic constraints, and every set $E \subseteq X$,

$$\dim^{\varphi}_{\mathrm{H}}(E) = \min_{A\subseteq\mathbb{N}} \sup_{x\in E} \dim^{\varphi,A}(x) \tag{1.3}$$

and

$$\dim^{\varphi}_{\mathrm{P}}(E) = \min_{A\subseteq\mathbb{N}} \sup_{x\in E} \mathrm{Dim}^{\varphi,A}(x). \tag{1.4}$$

Various nontrivial modifications to both machinery and proofs are necessary in getting from (1.1) and (1.2) to (1.3) and (1.4).

As an illustration of the power of our approach, we investigate the dimensions of hyperspaces. The *hyperspace* $\mathcal{K}(X)$ of a metric space X is the set of all nonempty compact subsets of X, equipped with the Hausdorff metric [44]. (For example, the "stages" $E_0, E_1, E_2, \ldots$ of a self-similar fractal $E \subseteq \mathbb{R}^n$ converge to E in the hyperspace $\mathbb{R}^n$.) The hyperspace of a separable metric space is itself a separable metric space, and the hyperspace is typically infinite-dimensional, even when the underlying metric space is finite-dimensional. One use of gauge families is reducing such infinite dimensions to enable quantitative comparisons. For example, McClure [30] defined, for each gauge family φ, a *jump* $\widetilde{\varphi}$ (our notation) that is also a gauge family, and he proved [31] for every *self-similar* subset E of a separable metric space X,

$$\dim_{\mathrm{H}}^{\widetilde{\theta}}(\mathcal{K}(E)) = \dim_{\mathrm{H}}(E),$$

where θ is the above-mentioned "un-gauged" gauge family.

Here we prove a *hyperspace dimension theorem* for the upper and lower Minkowski (i.e., box-counting) dimensions $\underline{\dim}_{\mathcal{M}}$ and $\overline{\dim}_{\mathcal{M}}$. This states that, for every separable metric space X, *every* gauge family φ, and *every* $E \subseteq X$,

$$\underline{\dim}_{\mathcal{M}}^{\widetilde{\varphi}}(\mathcal{K}(E)) = \underline{\dim}_{\mathcal{M}}^{\varphi}(E) \tag{1.5}$$

and

$$\overline{\dim}_{\mathcal{M}}^{\widetilde{\varphi}}(\mathcal{K}(E)) = \overline{\dim}_{\mathcal{M}}^{\varphi}(E). \tag{1.6}$$

We note that it is implicit in [30] that these identities hold for totally bounded sets E and gauge families φ satisfying a doubling condition.

Our third main result (Theorem 5.2) says that, for every separable metric space X, every "well-behaved" gauge family φ, and every compact set $E \subseteq X$,

$$\dim_{\mathrm{P}}^{\widetilde{\varphi}}(\mathcal{K}(E)) = \overline{\dim}_{\mathcal{M}}^{\widetilde{\varphi}}(\mathcal{K}(E)). \tag{1.7}$$

Our proof of this result makes essential use of (1.6) and the point-to-set principle (1.4).

Finally, we use the point-to-set principle (1.4), the identities (1.6) and (1.7), and some additional machinery to prove the *hyperspace packing dimension theorem* (Theorem 5.4), which says that, for every separable metric space X, every well-behaved gauge family φ, and every *analytic* (i.e., $\mathbf{\Sigma}_1^1$, an analog of NP that Sipser famously investigated [37, 38, 39]) set $E \subseteq X$,

$$\dim_{\mathrm{P}}^{\widetilde{\varphi}}(\mathcal{K}(E)) \geq \dim_{\mathrm{P}}^{\varphi}(E). \tag{1.8}$$

It is implicit in [30] that (1.8) holds for all σ-compact sets E.

At the time of this writing it is an open question whether there is an analogous hyperspace dimension theorem for Hausdorff dimension.

David Hilbert famously wrote the following [10].

> The final test of every new theory is its success in answering preexistent questions that the theory was not specifically created to answer.

The theory of algorithmic dimensions passed Hilbert's final test when the point-to-set principle gave us the results in the first paragraph of this introduction. We hope that the machinery developed here will lead to further such successes in the wider arena of separable metric spaces.

2 Gauged Classical Dimensions

We review the definitions of gauged Hausdorff, packing, and Minkowski dimensions. We refer the reader to [7, 28] for a complete introduction and motivation.

Let (X, ρ) be a metric space where ρ is the metric. (From now on we will omit ρ when referring to the space (X, ρ).) X is *separable* if there exists a countable set $D \subseteq X$ that is *dense* in X, meaning that for every $x \in X$ and $\delta > 0$, there is a $d \in D$ such that $\rho(x, d) < \delta$. The *diameter* of a set $E \subseteq X$ is $\operatorname{diam}(E) = \sup\{\rho(x, y) \mid x, y \in E\}$; notice that the diameter of a set can be infinite. A *cover* of $E \subseteq X$ is a collection $\mathcal{U} \subseteq \mathcal{P}(X)$ such that $E \subseteq \bigcup_{U \in \mathcal{U}} U$, and a δ-*cover* of E is a cover $\mathcal{U}$ of E such that $\operatorname{diam}(U) \leq \delta$ for all $U \in \mathcal{U}$.

▶ **Definition** (gauge functions and families)**.** A *gauge function* is a continuous,[5] nondecreasing function from $[0,\infty)$ to $[0,\infty)$ that vanishes only at 0 [8, 36]. A *gauge family* is a one-parameter family $\varphi = \{\varphi_s \,|\, s \in (0,\infty)\,\}$ of gauge functions φ_s satisfying

$$\varphi_s(\delta) = o(\varphi_t(\delta)) \text{ as } \delta \to 0^+$$

whenever $s > t$.

The *canonical gauge family* is $\theta = \{\theta_s \,|\, s \in (0,\infty)\,\}$, defined by $\theta_s(\delta) = \delta^s$. "Un-gauged" or "ordinary" Hausdorff, packing, and Minkowski dimensions are special cases of the following definitions, using $\varphi = \theta$.

Some of our gauged dimension results will require the existence of a "precision family" for the gauge family.

▶ **Definition** (precision family)**.** A *precision sequence* for a gauge function φ is a function $\alpha : \mathbb{N} \to \mathbb{Q}^+$ that vanishes as $r \to \infty$ and satisfies $\varphi(\alpha(r)) = O(\varphi(\alpha(r+1)))$ as $r \to \infty$. A *precision family* for a gauge family $\varphi = \{\varphi_s \mid s \in (0,\infty)\}$ is a one-parameter family $\alpha = \{\alpha_s \mid s \in (0,\infty)\}$ of precision sequences satisfying

$$\sum_{r\in\mathbb{N}} \frac{\varphi_t(\alpha_s(r))}{\varphi_s(\alpha_s(r))} < \infty$$

whenever $s < t$.

▶ **Observation 2.1.** *$\alpha_s(r) = 2^{-sr}$ is a precision family for the canonical gauge family θ.*

▶ **Definition** (gauged Hausdorff measure and dimension)**.** For every metric space X, set $E \subseteq X$, and gauge function φ, the *φ-gauged Hausdorff measure* of E is

$$H^{\varphi}(E) = \lim_{\delta\to 0^+} \inf\left\{ \sum_{U\in\mathcal{U}} \varphi(\mathrm{diam}(U)) \,\middle|\, \mathcal{U} \text{ is a countable } \delta\text{-cover of } E \right\}.$$

For every gauge family $\varphi = \{\varphi_s \,|\, s \in (0,\infty)\,\}$, the *$\varphi$-gauged Hausdorff dimension* of E is

$$\dim_{\mathrm{H}}^{\varphi}(E) = \inf\{s \in (0,\infty) \,|\, H^{\varphi_s}(E) = 0\,\}.$$

▶ **Definition** (gauged packing measure and dimension)**.** For every metric space X, set $E \subseteq X$, and $\delta \in (0,\infty)$, let $\mathcal{V}_\delta(E)$ be the set of all countable collections of disjoint open balls with centers in E and diameters at most δ. For every gauge function φ and $\delta > 0$, define the quantity

$$P_\delta^{\varphi}(E) = \sup_{\mathcal{U}\in\mathcal{V}_\delta(E)} \sum_{U\in\mathcal{U}} \varphi(\mathrm{diam}(U)).$$

Then the *φ-gauged packing pre-measure* of E is

$$P_0^{\varphi}(E) = \lim_{\delta\to 0^+} P_\delta^{\varphi}(E),$$

[5] Some authors require only that the function is right-continuous when working with Hausdorff dimension and left-continuous when working with packing dimension. Indeed, left continuity is sufficient for our hyperspace packing dimension theorem.

and the φ-*gauged packing measure* of E is

$$P^{\varphi}(E) = \inf\left\{ \sum_{U\in\mathcal{U}} P^{\varphi}_0(U) \;\middle|\; \mathcal{U} \text{ is a countable cover of } E \right\}.$$

For every gauge family $\varphi = \{\varphi_s \,|\, s \in (0,\infty)\}$, the φ-*gauged packing dimension* of E is

$$\dim^{\varphi}_{\mathrm{P}}(E) = \inf\{s \in (0,\infty) \,|\, P^{\varphi_s}(E) = 0\}.$$

▶ **Definition** (gauged Minkowski dimensions)**.** For every metric space X, $E \subseteq X$, and $\delta \in (0,\infty)$, let

$$N(E,\delta) = \min\left\{ |F| \;\middle|\; F \subseteq X \text{ and } E \subseteq \bigcup_{x\in F} B_\delta(x) \right\},$$

where $B_\delta(x)$ is the open ball of radius δ centered at x. Then for every gauge family $\varphi = \{\varphi_s\}_{s\in(0,\infty)}$ the φ-*gauged lower* and *upper Minkowski dimension* of E are

$$\underline{\dim}^{\varphi}_{\mathcal{M}}(E) = \inf\left\{ s \;\middle|\; \liminf_{\delta\to 0^+} N(E,\delta)\varphi_s(\delta) = 0 \right\}$$

and

$$\overline{\dim}^{\varphi}_{\mathcal{M}}(E) = \inf\left\{ s \;\middle|\; \limsup_{\delta\to 0^+} N(E,\delta)\varphi_s(\delta) = 0 \right\},$$

respectively.

When X is separable, it is sometimes useful to require that the balls covering E have centers in the countable dense set D. For all $E \subseteq X$ and $\delta \in (0,\infty)$, let

$$\hat{N}(E,\delta) = \min\left\{ |F| \;\middle|\; F \subseteq D \text{ and } E \subseteq \bigcup_{x\in F} B_\delta(x) \right\}.$$

▶ **Observation 2.2.** *If X is a separable metric space and $\varphi = \{\varphi_s\}_{s\in(0,\infty)}$ is a gauge family, then for all $E \subseteq X$,*

1. $\underline{\dim}^{\varphi}_{\mathcal{M}}(E) = \inf\left\{ s \;\middle|\; \liminf_{\delta\to 0^+} \hat{N}(E,\delta)\varphi_s(\delta) = 0 \right\}.$
2. $\overline{\dim}^{\varphi}_{\mathcal{M}}(E) = \inf\left\{ s \;\middle|\; \limsup_{\delta\to 0^+} \hat{N}(E,\delta)\varphi_s(\delta) = 0 \right\}.$

The following relationship between upper Minkowski dimension and packing dimension was previously known to hold for the canonical gauge family θ, a result that is essentially due to Tricot [42]. Our proof of this gauged generalization, is adapted from the presentation by Bishop and Peres [3] of the un-gauged proof.

▶ **Lemma 2.3** (generalizing Tricot [42])**.** *Let X be any metric space, $E \subseteq X$, and φ a gauge family.*

1. *If $\varphi_t(2\delta) = O(\varphi_s(\delta))$ as $\delta \to 0^+$ for all $s < t$, then*

$$\dim^{\varphi}_{\mathrm{P}}(E) \geq \inf\left\{ \sup_{i\in\mathbb{N}} \overline{\dim}^{\varphi}_{\mathcal{M}}(E_i) \;\middle|\; E \subseteq \bigcup_{i\in\mathbb{N}} E_i \right\}.$$

2. *If there is a precision family for φ, then*

$$\dim^{\varphi}_{\mathrm{P}}(E) \leq \inf\left\{ \sup_{i\in\mathbb{N}} \overline{\dim}^{\varphi}_{\mathcal{M}}(E_i) \;\middle|\; E \subseteq \bigcup_{i\in\mathbb{N}} E_i \right\}.$$

3 Gauged Algorithmic Dimensions

In this section we formulate algorithmic dimensions in arbitrary separable metric spaces and with arbitrary gauge families.

For the rest of this paper, let $X = (X, \rho)$ be a separable metric space, and fix a function $f : \{0,1\}^* \to X$ such that the set $D = \mathrm{range}(f)$ is dense in X. The metric space X is *computable* if there is a computable function $g : (\{0,1\}^*)^2 \times \mathbb{Q}^+ \to \mathbb{Q}$ that approximates ρ on D in the sense that, for all $v, w \in \{0,1\}^*$ and $\delta \in \mathbb{Q}^+$.

$$|g(v,w,\delta) - \rho(f(v), f(w))| \leq \delta.$$

Our results here hold for all separable metric spaces, whether or not they are computable, but our methods make explicit use of the function f.

Following standard practice [33, 4, 19], fix a universal oracle Turing machine U, and define the *(plain) Kolmogorov complexity* of a string $w \in \{0,1\}^*$ *relative to* an oracle $A \subseteq \mathbb{N}$ to be

$$\mathrm{C}^A(w) = \min\left\{|\pi| \,\middle|\, \pi \in \{0,1\}^* \text{ and } U^A(\pi) = w\right\},$$

i.e., the minimum number of bits required to cause U to output w when it has access to the oracle A. The *(plain) Kolmogorov complexity* of w is then $\mathrm{C}(w) = \mathrm{C}^{\emptyset}(w)$.

We define the *(plain) Kolmogorov complexity* of a point $q \in D$ to be

$$\mathrm{C}(q) = \min\{\mathrm{C}(w) \mid w \in \{0,1\}^* \text{ and } f(w) = q\},$$

noting that this depends on the enumeration f of D that we have fixed.

The *Kolmogorov complexity* of a point $x \in X$ at *precision* $\delta \in (0, \infty)$ is

$$\mathrm{C}_\delta(x) = \min\{\mathrm{C}(q) \mid q \in D \text{ and } \rho(q,x) < \delta\}.$$

The *algorithmic dimension* of a point $x \in X$ is

$$\dim(x) = \liminf_{\delta \to 0^+} \frac{\mathrm{C}_\delta(x)}{\log(1/\delta)}, \tag{3.1}$$

and the *strong algorithmic dimension* of x is

$$\mathrm{Dim}(x) = \limsup_{\delta \to 0^+} \frac{\mathrm{C}_\delta(x)}{\log(1/\delta)}. \tag{3.2}$$

These two dimensions[6] have been extensively investigated in the special cases where X is a Euclidean space $\mathbb{R}^n$ or a sequence space Σ^ω [23, 4].

Having generalized algorithmic dimensions to arbitrary separable metric spaces, we now generalize them to arbitrary gauge families.

Let $\varphi = \{\varphi_s \mid s \in (0,\infty)\}$ be a gauge family. Then, the *φ-gauged algorithmic dimension* of a point $x \in X$ is

$$\dim^\varphi(x) = \inf\left\{s \,\middle|\, \liminf_{\delta \to 0^+} 2^{\mathrm{C}_\delta(x)} \varphi_s(\delta) = 0\right\}, \tag{3.3}$$

[6] The definitions given here differ slightly from the standard formulation in which prefix Kolmogorov complexity is used instead of plain Kolmogorov complexity and the precision parameter δ belongs to $\{2^{-r} \mid r \in \mathbb{N}\}$. The present formulation is equivalent to the standard one for un-gaugued dimensions and facilitates our generalization to gauged algorithmic dimensions. In particular, plain Kolmogorov complexity is only needed to accommodate gauge functions φ in which the convergence of φ to 0 as $\delta \to 0^+$ is very slow.

and the *φ-gauged strong algorithmic dimension* of x is

$$\mathrm{Dim}^{\varphi}(x) = \inf\left\{ s \,\middle|\, \limsup_{\delta\to 0^+} 2^{\mathrm{C}_\delta(x)}\varphi_s(\delta) = 0 \right\}, \tag{3.4}$$

Gauged algorithmic dimensions $\dim^{\varphi}(x)$ have been investigated by Staiger [41] in the special case where X is a sequence space Σ^ω.

A routine inspection of (3.1)–(3.4) verifies the following.

▶ **Observation 3.1.** *For all $x \in X$, $\dim^{\theta}(x) = \dim(x)$ and $\mathrm{Dim}^{\theta}(x) = \mathrm{Dim}(x)$, where θ is the canonical gauge family given by $\theta_s(\delta) = \delta^s$.*

A specific investigation of algorithmic (or classical) dimensions might call for a particular gauge function or family for one of two reasons. First, many gauge functions may assign the same dimension to an object under consideration (because they converge to 0 at somewhat similar rates as $\delta \to 0^+$) but additional considerations may identify one of these as being the most precisely tuned to the phenomenon of interest. Finding such a gauge function is called finding the "exact dimension" of the object under investigation. This sort of calibration has been studied extensively for classical dimensions [7, 36] and by Staiger [41] for algorithmic dimension.

The second reason, and the reason of interest to us here, why specific investigations might call for particular gauge families is that a given gauge family φ may be so completely out of tune with the phenomenon under investigation that the φ-gauged dimensions of the objects of interest are either all minimum (all 0) or else all maximum (all the same dimension as the space X itself). In such a circumstance, a gauge family that converges to 0 more quickly or slowly than φ may yield more informative dimensions. Several such circumstances were investigated in a complexity-theoretic setting by Hitchcock, J. Lutz, and Mayordomo [11].

The following routine observation indicates the direction in which one adjusts a gauge family's convergence to 0 in order to adjust the resulting gauged dimensions upward or downward.

▶ **Observation 3.2.** *If φ and ψ are gauge families with $\varphi_s(\delta) = o(\psi_s(\delta))$ as $\delta \to 0^+$ for all $s \in (0, \infty)$, then, for all $x \in X$, $\dim^{\varphi}(x) \le \dim^{\psi}(x)$ and $\mathrm{Dim}^{\varphi}(x) \le \mathrm{Dim}^{\psi}(x)$.*

We now define an operation on gauge families that is implicit in earlier work [30] and is explicitly used in the results of Section 5.

▶ **Definition** (jump)**.** The *jump* of a gauge family φ is the family $\widetilde{\varphi}$ given $\widetilde{\varphi}_s(\delta) = 2^{-1/\varphi_s(\delta)}$.

▶ **Observation 3.3.** *The jump of a gauge family is a gauge family.*

We now note that the jump of a gauge family always converges to 0 more quickly than the original gauge family.

▶ **Lemma 3.4.** *For all gauge families φ and all $s \in (0, \infty)$, $\widetilde{\varphi}_s(\delta) = o(\varphi_s(\delta))$ as $\delta \to 0^+$.*

Observation 3.3 and Lemma 3.4 immediately imply the following.

▶ **Corollary 3.5.** *For all gauge families φ and all $x \in X$, $\dim^{\widetilde{\varphi}}(x) \le \dim^{\varphi}(x)$ and $\mathrm{Dim}^{\widetilde{\varphi}}(x) \le \mathrm{Dim}^{\varphi}(x)$.*

The definitions and results of this section relativize to arbitrary oracles $A \subseteq \mathbb{N}$ in the obvious manner, so the Kolmogorov complexities $\mathrm{C}^A(q)$ and $\mathrm{C}^A_\delta(x)$ and the dimensions $\dim^A(x)$, $\mathrm{Dim}^A(x)$, $\dim^{\varphi,A}(x)$, and $\mathrm{Dim}^{\varphi,A}(x)$ are all well-defined and behave as indicated.

▶ **Observation 3.6.** *For all gauge families φ, all $x \in X$, and all $s > 0$,*

$$\log\left(2^{\mathrm{C}_\delta(x)}\widetilde{\varphi}_s(\delta)\right) = \frac{\mathrm{C}_\delta(x)\varphi_s(\delta) - 1}{\varphi_s(\delta)}.$$

The $\widetilde{\varphi}$-gauged algorithmic dimensions admit the following characterizations, the second of which is used in the proof of our hyperspace packing dimension theorem.

▶ **Theorem 3.7.** *For all gauge families φ and all $x \in X$, the following identities hold.*

1. $\dim^{\widetilde{\varphi}}(x) = \inf\left\{ s \;\middle|\; \liminf_{\delta\to 0^+} \mathrm{C}_\delta(x)\varphi_s(\delta) = 0\right\}$.
2. $\mathrm{Dim}^{\widetilde{\varphi}}(x) = \inf\left\{ s \;\middle|\; \limsup_{\delta\to 0^+} \mathrm{C}_\delta(x)\varphi_s(\delta) = 0\right\}$.

4 The General Point-to-Set Principle

We now show that the point-to-set principle of J. Lutz and N. Lutz [22] holds in arbitrary separable metric spaces and for gauged dimensions. The proofs of these theorems are more delicate and involved than those in [22]. This is partially due to the fact that the metric spaces here need not be finite-dimensional, and to the weak restrictions we place on the gauge family.

▶ **Theorem 4.1** (general point-to-set principle for Hausdorff dimension)**.** *For every separable metric space X, every gauge family φ, and every set $E \subseteq X$,*

$$\dim^{\varphi}_{\mathrm{H}}(E) \geq \min_{A\subseteq\mathbb{N}} \sup_{x\in E} \dim^{\varphi,A}(x).$$

Equality holds if there is a precision family for φ.

▶ **Theorem 4.2** (general point-to-set principle for packing dimension)**.** *Let X be any separable metric space, $E \subseteq X$, and φ a gauge family.*

1. *If $\varphi_t(2\delta) = O(\varphi_s(\delta))$ and $\varphi_s(\delta) = O(1/\log\log(1/\delta))$ as $\delta \to 0^+$ for all $s < t$, then*

$$\dim^{\varphi}_{\mathrm{P}}(E) \geq \min_{A\subseteq\mathbb{N}} \sup_{x\in E} \mathrm{Dim}^{\varphi,A}(x).$$

2. *If there is a precision family for φ, then*

$$\dim^{\varphi}_{\mathrm{P}}(E) \leq \min_{A\subseteq\mathbb{N}} \sup_{x\in E} \mathrm{Dim}^{\varphi,A}(x).$$

Proof of Theorem 4.2.

1. Assume that $\varphi_t(2\delta) = O\left(\varphi_s(\delta)\right)$ and $\varphi_s(\delta) = O(1/\log\log(1/\delta))$ hold for all $s < t$. It suffices to show that there exists $A \subseteq \mathbb{N}$ such that, for all $x \in E$,

$$\mathrm{Dim}^{\varphi,A}(x) \leq \dim^{\varphi}_{\mathrm{P}}(E). \tag{4.1}$$

Let $s > t > u > \dim^{\varphi}_{\mathrm{P}}(E)$. Since $u > \dim^{\varphi}_{\mathrm{P}}(E)$, Lemma 2.3 and our hypothesis on φ tell us that there is a cover $\{E_i\}_{i\in\mathbb{Z}^+}$ of E such that, for all $i \in \mathbb{Z}^+$,

$$\overline{\dim}^{\varphi}_{\mathcal{M}}(E_i) \leq u. \tag{4.2}$$

For each $i \in \mathbb{Z}^+$ and $\delta \in \mathbb{Q} \cap (0, 1)$, let $F(i, \delta) \subseteq D$ satisfy

$$|F(i,\delta)| = \hat{N}(E_i, \delta)$$

and

$$E_i \subseteq \bigcup_{d \in F(i,\delta)} B_\delta(d).$$

Define $h : \mathbb{Z}^+ \times \mathbb{Q} \cap (0,1) \to (\{0,1\}^*)^*$ by

$$h(i,\delta) = \big(w_{i,\delta,1}, \dots, w_{i,\delta,\hat{N}(E_i,\delta)}\big),$$

where, recalling that f is the function mapping bit strings onto the dense set D,

$$F(i,\delta) = \left\{ f\big(w_{i,\delta,1}\big), \dots, f\big(w_{i,\delta,\hat{N}(E_i,\delta)}\big) \right\}.$$

Let A be an oracle encoding h.
To prove (4.1), let $x \in E$. It suffices to show that

$$\lim_{\delta \to 0^+} 2^{\mathrm{C}^A_\delta(x)} \varphi_s(\delta) = 0.$$

For this, let $\varepsilon > 0$. It suffices to show that, for all sufficiently small $\delta \in \mathbb{Q}^+$,

$$\mathrm{C}^A_\delta(x) < \log \frac{\varepsilon}{\varphi_s(\delta)}. \tag{4.3}$$

For each $\delta \in \mathbb{Q} \cap (0,1)$, let $r(\delta) = \lceil \log \frac{1}{\delta} \rceil$ and $\delta' = 2^{-r(\delta)}$, so that $\frac{\delta}{2} < \delta' \leq \delta$. Since $s > t$, our hypothesis on φ tells us that there is a constant $a > 0$ such that, for all sufficiently small $\delta \in \mathbb{Q}^+$,

$$\frac{1}{\varphi_t(\delta')} \leq \frac{a}{\varphi_s(2\delta')} \leq \frac{a}{\varphi_s(\delta)}. \tag{4.4}$$

Since $t > u$, (4.2) tells us that, for all $i \in \mathbb{N}$,

$$\lim_{\delta \to 0^+} \hat{N}(E_i, \delta) \varphi_t(\delta) = 0.$$

Hence, for all $i \in \mathbb{N}$ and all sufficiently small $\delta \in \mathbb{Q}^+$,

$$\hat{N}(E_i,\delta)\varphi_t(\delta) < \frac{\varepsilon}{2a}. \tag{4.5}$$

In particular, then, (4.4) and (4.5) tell us that, for all sufficiently small $\delta \in \mathbb{Q}^+$,

$$\hat{N}(E_i,\delta') \leq \frac{\varepsilon}{2a\varphi_t(\delta')} \leq \frac{\varepsilon}{2\varphi_s(\delta)}. \tag{4.6}$$

For each $i, k \in \mathbb{Z}^+$ and $\delta \in \mathbb{Q} \cap (0,1)$, let $\pi \in \{0,1\}^*$ be a string that encodes i, $r(\delta)$, and k, with

$$|\pi| = \log k + O(\log i + \log r(\delta)).$$

Let M be an oracle Turing machine that, with oracle A and program π, outputs the string $w_{i,\delta',k}$ that is the k^{th} component of $h(i,\delta')$ (if there is one), where $\delta' = 2^{-r(\delta)}$. Let c_M be an optimality constant for M.
To see that (4.3) holds, choose $i \in \mathbb{Z}^+$ such that $x \in E_i$, and let $\delta \in \mathbb{Q} \cap (0,1)$. Let $\delta' = 2^{-r(\delta)}$, and choose $k \in \big\{1, \dots, \hat{N}(E_i,\delta')\big\}$ such that $x \in B_{\delta'}\ \big(f\big(w_{i,\delta',k}\big)\big)$. Then

$$f\big(w_{i,\delta',k}\big) \in D \cap B_{\delta'}(x) \subseteq D \cap B_\delta(x),$$

so (4.6) gives, for all sufficiently small $\delta \in \mathbb{Q}^+$,

$$\begin{aligned} \mathrm{C}^A_\delta(x) &\le \mathrm{C}^A\big(w_{i,\delta',k}\big) \\ &\le \mathrm{C}^A_M\big(w_{i,\delta',k}\big) + c_M \\ &\le |\pi| + c_M \\ &\le \log k + c_M + O(\log i + \log r(\delta)) \\ &\le \log \hat{N}(E_i, \delta') + O(\log i + \log r(\delta)) \\ &\le \log \frac{\varepsilon}{2\varphi_s(\delta)} + O(\log i + \log r(\delta)). \end{aligned}$$

Since i is a constant and, by our assumption, $\log r(\delta) \le \log(\log(1/\delta) + 1) = O(1/\varphi_t(\delta)) = o(1/\varphi_s(\delta))$, the second term vanishes as $\delta \to 0^+$, affirming (4.3).

2. Let $s > t > \sup_{x \in E} \mathrm{Dim}^{\varphi, A}(x)$. Then for each $x \in E$ and all sufficiently small $\delta \in \mathbb{Q}^+$, $\mathrm{C}^A_\delta(x) < \log(1/\varphi_t(\delta))$. For all $\delta \in \mathbb{Q}^+$, let

$$\mathcal{U}_\delta = \left\{ B_\delta(f(w)) \;\middle|\; \mathrm{C}^A(w) \le \log(1/\varphi_t(\delta)) \right\},$$

and for each $i \in \mathbb{N}$, let

$$E_i = \{ x \mid \forall \delta < 1/i,\ x \in \mathcal{U}_\delta \}.$$

Then $E \subseteq \bigcup_{i \in \mathbb{N}} E_i$. For each $\delta < 1/i$, $N(E_i, \delta) < 2/\varphi_t(\delta)$, so $N(E_i, \delta)\varphi_s(\delta) = o(1)$, and therefore $\overline{\dim}^{\varphi}_{\mathcal{M}}(E_i) \le s$. Assuming that there is a precision family for φ, the result follows by Lemma 2.3. ◀

5 Hyperspace Dimension Theorems

This section presents our main theorems.

As before, let $X = (X, \rho)$ be a separable metric space. The *hyperspace* of X is the metric space $\mathcal{K}(X) = (\mathcal{K}(X), \rho_{\mathrm{H}})$, where $\mathcal{K}(X)$ is the set of all nonempty compact subsets of X and ρ_{H} is the *Hausdorff metric* [9] on $\mathcal{K}(X)$ defined by

$$\rho_{\mathrm{H}}(E, F) = \max\left\{ \sup_{x \in E} \rho(x, F), \sup_{y \in F} \rho(E, y) \right\},$$

where $\rho(x, F) = \inf_{y \in F} \rho(x, y)$ and $\rho(E, y) = \inf_{x \in E} \rho(x, y)$.

Let $f : \{0,1\}^* \to X$ and $D = \mathrm{range}(f)$ be fixed as at the beginning of section 3, so that D is dense in X. Let $\mathcal{D}$ be the set of all nonempty, finite subsets of D. It is well known and easy to show that $\mathcal{D}$ is a countable dense subset of $\mathcal{K}(X)$, and it is routine to define from f a function $\widetilde{f} : \{0,1\}^* \to \mathcal{K}(X)$ such that $\mathrm{range}(\widetilde{f}) = \mathcal{D}$. Thus $\mathcal{K}(X)$ is a separable metric space, and the results in section 4 hold for $\mathcal{K}(X)$.

It is important to note the distinction between the classical Hausdorff and packing dimensions $\dim_{\mathrm{H}}(E)$ and $\dim_{\mathrm{P}}(E)$ of a nonempty compact subset E of X and the algorithmic dimensions $\dim(E)$ and $\mathrm{Dim}(E)$ of this same set when it is regarded as a point in $\mathcal{K}(X)$.

Our first hyperspace dimension theorem applies to lower and upper Minkowski dimensions. This theorem, which is proven using a counting argument, is very general, placing no restrictions on the gauge family φ or the separable metric space X.

▶ **Theorem 5.1** (hyperspace Minkowski dimension theorem). *For every gauge family φ and every $E \subseteq X$,*

$$\underline{\dim}^{\widetilde{\varphi}}_{\mathcal{M}}(\mathcal{K}(E)) = \underline{\dim}^{\varphi}_{\mathcal{M}}(E) \quad \textit{and} \quad \overline{\dim}^{\widetilde{\varphi}}_{\mathcal{M}}(\mathcal{K}(E)) = \overline{\dim}^{\varphi}_{\mathcal{M}}(E).$$

Our third main result is the surprising fact that in a hyperspace, packing dimension and upper Minkowski dimension are equivalent for compact sets.

▶ **Theorem 5.2.** *For every separable metric space X, every compact set $E \subseteq X$, and every gauge family φ such that $\varphi_t(2\delta) = O(\varphi_s(\delta))$ and $\varphi_s(\delta) = O(1/\log\log(1/\delta))$ as $\delta \to 0^+$ for all $s < t$ and there is a precision family for φ,*

$$\dim_{\mathrm{P}}^{\widetilde{\varphi}}(\mathcal{K}(E)) = \overline{\dim}_{\mathcal{M}}^{\widetilde{\varphi}}(\mathcal{K}(E)).$$

The point-to-set principle is central to our proof of this theorem: We recursively construct a single compact set $L \subseteq E$ (i.e., a single point in the hyperspace $\mathcal{K}(E)$) that has high Kolmogorov complexity at infinitely many precisions, relative to an appropriate oracle A. We then invoke Theorem 3.7 to show that this L has high φ-gauged strong algorithmic dimension relative to A. By the point-to-set principle, then, $\mathcal{K}(E)$ has high packing dimension.

▶ **Observation 5.3.** *The conclusion of Theorem 5.2 does not hold for arbitrary sets E.*

Proof. Let $E = \{1/n : n \in \mathbb{N}\}$. Then $\overline{\dim}_{\mathcal{M}}^{\theta}(E) = 1/2$, but every compact subset of E is finite, so $\mathcal{K}(E)$ is countable and $\dim_{\mathrm{P}}^{\widetilde{\theta}}(\mathcal{K}(E)) = 0$. ◀

▶ **Theorem 5.4** (hyperspace packing dimension theorem)**.** *If X is a separable metric space, $E \subseteq X$ is an analytic set, and φ is a gauge family such that $\varphi_s(2\delta) = O(\varphi_s(\delta))$ and $\varphi_s(\delta) = O(1/\log\log(1/\delta))$ as $\delta \to 0^+$ for all $s \in (0, \infty)$ and there is a precision family for φ, then*

$$\dim_{\mathrm{P}}^{\widetilde{\varphi}}(\mathcal{K}(E)) \geq \dim_{\mathrm{P}}^{\varphi}(E).$$

Proof. For compact sets E, Theorem 5.2 and the hyperspace Minkowski dimension theorem (Theorem 5.1) imply $\dim_{\mathrm{P}}^{\widetilde{\varphi}}(\mathcal{K}(E)) = \overline{\dim}_{\mathcal{M}}^{\varphi}(E)$.

A result of Joyce and Preiss (Corollary 1 in [13]) states that every analytic set with positive (possibly infinite) gauged packing measure contains a compact subset with positive (finite) packing measure in the same gauge. It follows that if E is analytic, then for all $\varepsilon > 0$ there exists a compact subset $E_\varepsilon \subseteq E$ with $\dim_{\mathrm{P}}^{\varphi}(E_\varepsilon) \geq \dim_{\mathrm{P}}^{\varphi}(E) - \varepsilon$. Therefore

$$\begin{aligned}\dim_{\mathrm{P}}^{\widetilde{\varphi}}(\mathcal{K}(E_\varepsilon)) &= \overline{\dim}_{\mathcal{M}}^{\varphi}(E_\varepsilon)\\ &\geq \dim_{\mathrm{P}}^{\varphi}(E_\varepsilon)\\ &\geq \dim_{\mathrm{P}}^{\varphi}(E) - \varepsilon.\end{aligned}$$

Letting $\varepsilon \to 0$ completes the proof. ◀

6 Conclusion

Our results exhibit and amplify the power of the theory of computing to make unexpected contributions to other areas of the mathematical sciences. We hope and expect to see more such results in the near future.

We mention three open problems whose solutions may contribute to such progress. First, at the time of this writing, a hyperspace Hausdorff dimension theorem remains an open problem. The difficulty in adapting our approach to that problem is that in the proof of Theorem 5.2, the set L we construct is only guaranteed to have high complexity at infinitely many precisions. An analogous proof for Hausdorff dimension would require constructing a set L that has high complexity at all but finitely many precisions.

Second, it would be useful to identify classes of spaces in which Billingsley-type algorithmic dimensions – dimensions shaped by probability measures – can be formulated.

Finally, we do not at this time know how to characterize algorithmic dimensions in separable metric spaces in terms of martingales or more general gales. This is despite the fact that algorithmic dimensions were first formulated in these terms in sequence spaces [20, 1].

References

1 Krishna B. Athreya, John M. Hitchcock, Jack H. Lutz, and Elvira Mayordomo. Effective strong dimension in algorithmic information and computational complexity. *SIAM Journal on Computing*, 37(3):671–705, 2007.

2 Patrick Billingsley. *Ergodic Theory and Information.* John Wiley & Sons, 1965.

3 Christopher J. Bishop and Yuval Peres. *Fractals in Probability and Analysis.* Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2016. doi:10.1017/9781316460238.

4 Rod G. Downey and Denis R. Hirschfeldt. *Algorithmic Randomness and Complexity.* Springer-Verlag, 2010.

5 Rod G. Downey and Denis R. Hirschfeldt. Algorithmic randomness. *Communications of the ACM*, 62(5):70–80, 2019.

6 Rod G. Downey and Denis R. Hirschfeldt. Computability and randomness. *Notices of the American Mathematical Society*, 66(7):1001–1012, 2019.

7 Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications, 3rd edition.* John Wiley & Sons, 2014.

8 Felix Hausdorff. Dimension und äußeres Maß. *Math. Ann.*, 79:157–179, 1919. English translation: Dimension and Outer Measure. In Gerald A. Edgar, editor, *Classics on Fractals*, pages 75–100. Addison-Wesley, 1993.

9 Felix Hausdorff. *Grundzüge der Mengenlehre.* AMS Chelsea Publishing, 2005. Leipzig, 1914. English translation: Felix Hausdorff, *Set Theory*, AMS Chelsea Publishing, 2005.

10 David Hilbert. On the infinite. In Jean van Heijenoort, editor, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931.* Harvard University Press, 1967. Translation by Stefan Bauer-Mengelberg of Hilbert's 1925 essay.

11 John M. Hitchcock, Jack H. Lutz, and Elvira Mayordomo. Scaled dimension and nonuniform complexity. *Journal of Computer and System Sciences*, 69:97–122, 2004.

12 Greg Hjorth and Alexander S. Kechris. New dichotomies for Borel equivalence relations. *Bull. Symbolic Logic*, 3(3):329–346, 1997. doi:10.2307/421148.

13 H. Joyce and D. Preiss. On the existence of subsets of finite positive packing measure. *Mathematika*, 42(1):15–24, 1995.

14 Anatole Katok and Boris Hasselblatt. *Introduction to the Modern Theory of Dynamical Systems.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995.

15 Nets Hawk Katz and Terence Tao. Some connections between Falconer's distance set conjecture and sets of Furstenburg type. *New York Journal of Mathematics*, 7:149–187, 2001.

16 Robert Kaufman. On Hausdorff dimension of projections. *Mathematika*, 15(2):153–155, 1968.

17 Alexander S. Kechris, Slawomir Solecki, and Stevo Todorcevic. Borel chromatic numbers. *Adv. Math.*, 141(1):1–44, 1999. doi:10.1006/aima.1998.1771.

18 Takayuki Kihara and Arno Pauly. Point degree spectra of represented spaces. *CoRR*, abs/1405.6866, 2014. arXiv:1405.6866.

19 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, Berlin, 2019. Fourth Edition.

20 Jack H. Lutz. The dimensions of individual strings and sequences. *Information and Computation*, 187(1):49–79, 2003.

21 Jack H. Lutz. The point-to-set principle, the Continuum Hypothesis, and the dimensions of Hamel bases. Technical report, arXiv, 2020. arXiv:2109.10981.

22 Jack H. Lutz and Neil Lutz. Algorithmic information, plane Kakeya sets, and conditional dimension. *ACM Transactions on Computation Theory*, 10(2):7:1–7:22, 2018.

23 Jack H. Lutz and Neil Lutz. Who asked us? How the theory of computing answers questions about analysis. In Dingzhu Du and Jie Wang, editors, *Complexity and Approximation: In Memory of Ker-I Ko*, pages 48–56. Springer, 2020.

24 Jack H. Lutz and Elvira Mayordomo. Algorithmic fractal dimensions in geometric measure theory. In Vasco Brattka and Peter Hertling, editors, *Handbook of Computability and Complexity in Analysis*. Springer, 2021.

25 Neil Lutz. Fractal intersections and products via algorithmic dimension. *ACM Trans. Comput. Theory*, 13(3), 2021.

26 Neil Lutz and D. M. Stull. Projection theorems using effective dimension. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27–31, 2018, Liverpool, UK*, pages 71:1–71:15, 2018.

27 Neil Lutz and D.M. Stull. Bounding the dimension of points on a line. *Information and Computation*, 275, 2020.

28 Pertti Mattila. *Geometry of Sets and Measures in Euclidean Spaces: Fractals and Rectifiability*. Cambridge University Press, 1995.

29 Elvira Mayordomo. Effective Hausdorff dimension in general metric spaces. *Theory of Computing Systems*, 62:1620–1636, 2018.

30 Mark McClure. Entropy dimensions of the hyperspace of compact sets. *Real Anal. Exchange*, 21(1):194–202, 1995. URL: `https://projecteuclid.org:443/euclid.rae/1341343235`.

31 Mark McClure. The Hausdorff dimension of the hyperspace of compact sets. *Real Anal. Exchange*, 22:611–625, 1996.

32 Yiannis N. Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing, 1980.

33 Andre Nies. *Computability and Randomness*. Oxford University Press, 2009.

34 Tuomas Orponen. Combinatorial proofs of two theorems of Lutz and Stull. *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 1–12, 2021.

35 Yakov Pesin. *Dimension Theory in Dynamical Systems: Contemporary Views and Applications*. University of Chicago Press, 1998.

36 Claude A. Rogers. *Hausdorff Measures*. Cambridge University Press, 1998. Originally published in 1970.

37 M. Sipser. A complexity-theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

38 M. Sipser. A topological view of some problems in complexity theory. In *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science*, pages 567–572, 1984.

39 M. Sipser. The history and status of the P versus NP question. In *Proceedings of the 24th Annual ACM Symposium on the theory of Computing*, pages 603–618, 1992.

40 Theodore Slaman. Kolmogorov complexity and capacitability of dimension. In *Computability Theory (hybrid meeting), Report No. 21/2021, Mathematisches Forschungsinstitut Oberwolfach*, 2021.

41 Ludwig Staiger. Exact constructive and computable dimensions. *Theory Comput. Syst.*, 61(4):1288–1314, 2017.

42 Claude Tricot. Two definitions of fractional dimension. *Mathematical Proceedings of the Cambridge Philosophical Society*, 91:57–74, 1982.

43 Shmuel Weinberger. *Computers, Rigidity, and Moduli: The Large-Scale Fractal Geometry of Riemannian Moduli Space*. Princeton University Press, Princeton, NJ, USA, 2004.

44 Stephen Willard. *General Topology*. Dover Publications, 2004.

One-Way Communication Complexity and Non-Adaptive Decision Trees

Nikhil S. Mande
CWI, Amsterdam, The Netherlands

Swagato Sanyal
Indian Institute of Technology, Kharagpur, India

Suhail Sherif
Vector Institute, Toronto, Canada

Abstract

We study the relationship between various one-way communication complexity measures of a composed function with the analogous decision tree complexity of the outer function. We consider two gadgets: the AND function on 2 inputs, and the Inner Product on a constant number of inputs. More generally, we show the following when the gadget is Inner Product on $2b$ input bits for all $b \geq 2$, denoted IP.

- If f is a total Boolean function that depends on all of its n input bits, then the bounded-error one-way quantum communication complexity of $f \circ \mathsf{IP}$ equals $\Omega(n(b-1))$.
- If f is a *partial* Boolean function, then the deterministic one-way communication complexity of $f \circ \mathsf{IP}$ is at least $\Omega(b \cdot \mathsf{D}_{\mathrm{dt}}^{\rightarrow}(f))$, where $\mathsf{D}_{\mathrm{dt}}^{\rightarrow}(f)$ denotes non-adaptive decision tree complexity of f.

To prove our quantum lower bound, we first show a lower bound on the VC-dimension of $f \circ \mathsf{IP}$. We then appeal to a result of Klauck [STOC'00], which immediately yields our quantum lower bound. Our deterministic lower bound relies on a combinatorial result independently proven by Ahlswede and Khachatrian [Adv. Appl. Math.'98], and Frankl and Tokushige [Comb.'99].

It is known due to a result of Montanaro and Osborne [arXiv'09] that the deterministic one-way communication complexity of $f \circ \mathsf{XOR}$ *equals* the non-adaptive parity decision tree complexity of f. In contrast, we show the following when the inner gadget is the AND function on 2 input bits.

- There exists a function for which even the *quantum* non-adaptive AND decision tree complexity of f is exponentially large in the deterministic one-way communication complexity of $f \circ \mathsf{AND}$.
- However, for symmetric functions f, the non-adaptive AND decision tree complexity of f is at most quadratic in the (even two-way) communication complexity of $f \circ \mathsf{AND}$.

In view of the first bullet, a lower bound on non-adaptive AND decision tree complexity of f *does not* lift to a lower bound on one-way communication complexity of $f \circ \mathsf{AND}$. The proof of the first bullet above uses the well-studied *Odd-Max-Bit* function. For the second bullet, we first observe a connection between the one-way communication complexity of f and the *Möbius sparsity* of f, and then give a lower bound on the Möbius sparsity of symmetric functions. An upper bound on the non-adaptive AND decision tree complexity of symmetric functions follows implicitly from prior work on combinatorial group testing; for the sake of completeness, we include a proof of this result.

It is well known that the rank of the communication matrix of a function F is an upper bound on its deterministic one-way communication complexity. This bound is known to be tight for some F. However, in our final result we show that this is not the case when $F = f \circ \mathsf{AND}$. More precisely we show that for all f, the deterministic one-way communication complexity of $F = f \circ \mathsf{AND}$ is at most $(\mathrm{rank}(M_F))(1 - \Omega(1))$, where M_F denotes the communication matrix of F.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Oracles and decision trees

Keywords and phrases Decision trees, communication complexity, composed Boolean functions

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.49

Related Version *Full Version*: https://arxiv.org/abs/2105.01963

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 49; pp. 49:1–49:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Nikhil S. Mande*: Supported by the Dutch Research Council (NWO) through QuantERA project QuantAlgo 680-91-034.
Swagato Sanyal: Supported by an ISIRD Grant from Sponsored Research and Industrial Consultancy, IIT Kharagpur. Part of this work was done while the author was at NTU and CQT, and supported by the Singapore National Research Foundation under NRF RF Award No. NRF-NRFF2013-13.

Acknowledgements Swagato Sanyal thanks Prahladh Harsha and Jaikumar Radhakrishnan for pointing out the reference [15], and Srijita Kundu for pointing out the reference [26]. N.S.M. thanks Ronald de Wolf for useful discussions. We thank Justin Thaler for pointing out a bug from an earlier version of the paper, and Arnab Maiti for helpful comments. We thank anonymous reviewers for helpful comments and suggestions, including an improvement of the value b from 4 to 2 in Theorem 1.3.

1 Introduction

Composed functions are important objects of study in analysis of Boolean functions and computational complexity. For Boolean functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, their composition $f \circ g : (\{0,1\}^m)^n \to \{0,1\}$ is defined as follows: $f \circ g(x_1, \ldots, x_n) := f(g(x_1), \ldots, g(x_n))$. In other words, $f \circ g$ is the function obtained by first computing g on n disjoint inputs of m bits each, and then computing f on the n resultant bits. Composed functions have been extensively looked at in the complexity theory literature, with respect to various complexity measures [8, 24, 39, 42, 43, 9, 44, 35, 5, 18, 2, 17, 4].

Of particular interest to us is the case when g is a communication problem (also referred to as "gadget"). More precisely, let $g : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ and $f : \{0,1\}^n \to \{0,1\}$ be Boolean functions. Consider the following communication problem: Alice has input $x = (x_1, \ldots, x_n)$ and Bob has input $y = (y_1 \ldots, y_n)$ where $x_i, y_i \in \{0,1\}^b$ for all $i \in [n]$. Their goal is to compute $f \circ g((x_1, y_1), \ldots, (x_n, y_n))$ using as little communication as possible. A natural protocol is the following: Alice and Bob jointly simulate an efficient query algorithm for f, using an optimal communication protocol for g to answer each query. Lifting theorems are statements that say this naive protocol is essentially optimal. Such theorems enable us to prove lower bounds on the rich model of communication complexity by proving feasibly easier-to-prove lower bounds in the query complexity (decision tree) model. Various lifting theorems have been proved in the literature [19, 13, 38, 20, 11, 48, 16, 21, 22, 27, 30, 10].

In this work we are interested in the *one-way* communication complexity of composed functions. Here, a natural protocol is for Alice and Bob to simulate a *non-adaptive* decision tree for the outer function, using an optimal *one-way* communication protocol for the inner function. Thus, the one-way communication complexity of $f \circ g$ is at most the non-adaptive decision tree complexity of f times the one-way communication complexity of g.

Lifting theorems in the one-way model are less studied than in the two-way model. Montanaro and Osborne [36] observed that the deterministic one-way communication complexity of $f \circ \mathsf{XOR}$ *equals* the non-adaptive parity decision tree complexity of f. Thus, non-adaptive parity decision tree complexity lifts "perfectly" to deterministic communication complexity with the XOR gadget. Kannan et al. [25] showed that under uniformly distributed inputs, bounded-error non-adaptive parity decision tree complexity lifts to one-way bounded-error distributional communication complexity with the XOR gadget. Hosseini, Lovett and Yaroslavtsev [23] showed that randomized non-adaptive parity decision tree complexity lifts to randomized communication complexity with the XOR gadget in the one-way broadcasting model with $\Theta(n)$ players.

We explore the tightness of the naive communication upper bound for two different choices of the gadget g: the Inner Product function, and the two-input AND function. For each choice of g, we compare the one-way communication complexity of $f \circ g$ with an appropriate type of non-adaptive decision tree complexity of f. Below, we motivate and state our results for each choice of the gadget. Formal definitions of the measures considered in this section can be found in Section 2 and Appendix A.

Inner Product Gadget

Let $\mathsf{Q}^{\rightarrow}_{\mathrm{cc},\varepsilon}(\cdot)$, $\mathsf{R}^{\rightarrow}_{\mathrm{cc},\varepsilon}(\cdot)$ and $\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(\cdot)$ denote quantum ε-error, randomized ε-error and deterministic one-way communication complexity, respectively. When we allow the parties to share an arbitrary input-independent entangled state in the beginning of the protocol, denote the one-way quantum ε-error communication complexity by $\mathsf{Q}^{*,\rightarrow}_{\mathrm{cc},\varepsilon}(\cdot)$. Let $\mathsf{Q}^{\rightarrow}_{\mathrm{dt}}(\cdot)$ and $\mathsf{D}^{\rightarrow}_{\mathrm{dt}}(\cdot)$ denote bounded-error quantum non-adaptive decision tree complexity and deterministic non-adaptive decision tree complexity, respectively. For an integer $b > 0$, let $\mathsf{IP} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ denote the *Inner Product Modulo 2* function, that outputs the parity of the bitwise AND of two b-bit input strings. Our first result shows that if f is a total function that depends on all of its input bits, the quantum (and hence, randomized) bounded-error one-way communication complexity of $f \circ \mathsf{IP}$ is $\Omega(n(b-1))$. Let $\mathbb{H}_{\mathsf{bin}}(\cdot)$ denote the binary entropy function. If $\varepsilon = 1/2 - \Omega(1)$, then $1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon) = \Omega(1)$.

▶ **Theorem 1.1.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a total Boolean function that depends on all its inputs (i.e., it is not a junta on a strict subset of its inputs), and let $\varepsilon \in (0,1/2)$. Let $\mathsf{IP} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ denote the Inner Product function on $2b$ input bits for $b \geq 1$. Then $\mathsf{Q}^{\rightarrow}_{\mathrm{cc},\varepsilon}(f \circ \mathsf{IP}) \geq (1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon))n(b-1)$ and $\mathsf{Q}^{*,\rightarrow}_{\mathrm{cc},\varepsilon}(f \circ \mathsf{IP}) \geq (1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon))n(b-1)/2$.*

▶ Remark 1.2. In an earlier manuscript [40], the second author proved a lower bound of $(1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon))n(b-1)$ on a weaker complexity measure, namely $\mathsf{R}^{\rightarrow}_{\mathrm{cc},\varepsilon}(F)$, via information-theoretic tools. Kundu [28] subsequently observed that a quantum lower bound can also be obtained by additionally using Holevo's theorem. They also suggested to the second author via private communication that one might be able to recover these bounds using a result of Klauck [26]. This is indeed the approach we take, and we thank them for suggesting this and pointing out the reference.

In order to prove Theorem 1.1, we appeal to a result of Klauck [26, Theorem 3], who showed that the one-way ε-error quantum communication complexity of a function F is at least $(1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon)) \cdot \mathsf{VC}(F)$, where $\mathsf{VC}(F)$ denotes the VC-dimension of F (see Definition 2.7). In the case when the parties can share an arbitrary entangled state in the beginning of a protocol, Klauck showed a lower bound of $(1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon)) \cdot \mathsf{VC}(F)/2$. We exhibit a set of inputs that witnesses the fact that $\mathsf{VC}(f \circ \mathsf{IP}) \geq n(b-1)$. Note that Theorem 1.1 is useful only when $b > 1$. Indeed, no non-trivial lifting statement is true for $b = 1$ when f is the AND function on n bits, since in this case, $f \circ \mathsf{IP} = \mathsf{AND}_{2n}$, whose one-way communication complexity is 1.

Our second result with the Inner Product gadget relates the deterministic one-way communication complexity of $f \circ \mathsf{IP}$ to the deterministic non-adaptive decision tree complexity of f, where f is an arbitrary *partial* Boolean function.

▶ **Theorem 1.3.** *Let $\mathsf{S} \subseteq \{0,1\}^n$ be arbitrary, and $f : \mathsf{S} \to \{0,1\}$ be a* partial *Boolean function. Let $b \geq 2$ and $\mathsf{IP} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$. Then $\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{IP}) = \Omega(b \cdot \mathsf{D}^{\rightarrow}_{\mathrm{dt}}(f))$.*

Given a protocol Π, our proof extracts a set of variables of cardinality linear in the complexity of Π, whose values always determine the value of f. The following claim which follows directly from a result due to Ahlswede and Khachatrian [1] and independently Frankl and Tokushige [15], is a crucial ingredient in our proof.

▶ **Theorem 1.4.** *Let $q \geq 3$ and $1 \leq d \leq n/3$. Let $\mathcal{A} \subseteq [q]^n$ be such that for all $x^{(1)} = (x_1^{(1)}, \ldots, x_n^{(1)})$, $x^{(2)} = (x_1^{(2)}, \ldots, x_n^{(2)}) \in \mathcal{A}$, $|\{i \in [n] \mid x_i^{(1)} = x_i^{(2)}\}| \geq d$. Then, $|\mathcal{A}| < q^{n-\frac{d}{10}}$.*

We refer the reader to the full version of our paper [33] for a proof.

▶ Remark 1.5. An analogous lifting theorem for deterministic one-way protocols for *total* outer functions follows as a special case of both Theorem 1.1 and Theorem 1.3. However, the statement admits a simple and direct proof based on a fooling set argument.

Theorem 1.1 and Theorem 1.3 give lower bounds even when the gadget is the Inner Product function on 4 input bits (and lower bounds do not hold for the Inner Product gadget with fewer inputs). It is worth mentioning here that prior works that consider lifting theorems with the Inner Product gadget [11, 48, 10], albeit in the two-way model of communication complexity, require a super-constant gadget size.

AND Gadget

Interactive communication complexity of functions of the form $f \circ \mathsf{AND}$ have gained a recent interest [27, 47]. In order to state and motivate our results regarding when the inner gadget is the 2-bit AND function, we first discuss some known results in the case when the inner gadget is the 2-bit XOR function.

Consider non-adaptive decision trees, where the trees are allowed to query arbitrary parities of the input variables. Denote the minimum cost (number of parity queries) of such a tree computing a Boolean function f, by $\mathsf{NAPDT}(f)$. An efficient non-adaptive parity decision tree for f can easily be simulated to obtain an efficient deterministic one-way communication protocol for $f \circ \mathsf{XOR}$. Thus, $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{XOR}) \leq \mathsf{NAPDT}(f)$. Montanaro and Osborne [36] observed that this inequality is, in fact, tight for all Boolean functions. More precisely,

▷ Claim 1.6 ([36]). For all Boolean functions $f : \{0,1\}^n \to \{0,1\}$, $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{XOR}) = \mathsf{NAPDT}(f)$.

If the inner gadget were AND instead of XOR, then the natural analogous decision tree model to consider would be non-adaptive decision trees that have query access to arbitrary ANDs of subsets of inputs. Denote the minimum cost (number of AND queries) of such a tree computing a Boolean function f by $\mathsf{NAADT}(f)$. Clearly, $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND})$ is bounded from above by $\mathsf{NAADT}(f)$, since a non-adaptive AND decision tree can be easily simulated to give a one-way communication protocol for $f \circ \mathsf{AND}$ of the same complexity. Thus, $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \leq \mathsf{NAADT}(f)$. On the other hand, one can show that $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \geq \log(\mathsf{NAADT}(f))$ (see Claim 4.3). Thus

$$\log(\mathsf{NAADT}(f)) \leq \mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \leq \mathsf{NAADT}(f). \tag{1}$$

We explore if an analogous statement to Claim 1.6 holds true if the inner function were AND instead of XOR. That is, is the second inequality in Equation (1) always tight?

We give a negative answer in a very strong sense and exhibit a function for which the first inequality is tight (up to an additive constant). We show that there is an exponential separation between these measures even if one allows the decision trees to have *quantum* query access to ANDs of subsets of input variables. It is worth noting that, in contrast, if one is given quantum query access to *parities* (in place of ANDs) of subsets of input variables, then one can completely recover an n-bit string using just 1 query [6], rendering this model trivial. Let $\mathsf{QNAADT}(f)$ denote the bounded-error quantum non-adaptive AND decision tree complexity of f.

▶ **Theorem 1.7.** *There exists a function* $f : \{0,1\}^n \to \{0,1\}$ *such that* $\mathsf{QNAADT}(f) = \Omega(2^{\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND})})$.

The function f we use to witness the bound in Theorem 1.7 is a modification of the well-studied *Odd-Max-Bit* function, which we denote OMB_n. This function outputs 1 if and only if the maximum index of the input string that contains a 0, is odd (see Definition 2.3). A $\lceil \log(n+1) \rceil$-cost one-way communication protocol is easy to show, since Alice can simply send Bob the maximum index where her input is 0 (if it exists), and Bob can use this along with his input to conclude the parity of the maximum index where the bitwise AND of their inputs is 0. A crucial property that we use to show a lower bound of $\Omega(n)$ on $\mathsf{QNAADT}(\mathsf{OMB}_n)$ is that OMB_n has large *alternating number*, that is, there is a monotone path on the Boolean hypercube from 0^n to 1^n on which the value of OMB_n flips many times.

Theorem 1.7 implies that, in contrast to the lifting theorem with the XOR gadget (Claim 1.6), the measure of non-adaptive AND decision tree complexity *does not* lift to a one-way communication lower bound for $f \circ \mathsf{AND}$. However we show that a statement analogous to Claim 1.6 does hold true for symmetric functions f, albeit with a quadratic factor, even when the measure is two-way communication complexity, denoted $\mathsf{D}_{\mathrm{cc}}(\cdot)$.

▶ **Theorem 1.8.** *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a symmetric function. Then* $\mathsf{NAADT}(f) = O(\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND})^2)$.

In fact we prove a stronger bound in which $\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND})$ above is replaced by $\log \operatorname{rank}(M_{f \circ \mathsf{AND}})$, where $M_{f \circ \mathsf{AND}}$ denotes the communication matrix of $f \circ \mathsf{AND}$. That is, we show that for symmetric functions f,

$$\mathsf{NAADT}(f) = O(\log^2 \operatorname{rank}(M_{f \circ \mathsf{AND}})). \tag{2}$$

Since it is well known (Equation (6)) that the communication complexity of a function is at least as large as the logarithm of the rank of its communication matrix, this implies Theorem 1.8. There have been multiple works (see, for example, [8, 47, 27] and the references therein) studying the communication complexity of AND functions in connection with the log-rank conjecture [31] which states that the communication complexity is bounded from above by a polynomial in the logarithm of the rank of the communication matrix. Among other things, Buhrman and de Wolf [8] observed that the log-rank conjecture holds for symmetric functions composed with AND. In particular, they showed that if f is symmetric, then $\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND}) = O(\log \operatorname{rank}(M_{f \circ \mathsf{AND}})))$. Most recently, Knop et al. [27] showed that $\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND}) = O(\operatorname{poly}(\log \operatorname{rank}(M_{f \circ \mathsf{AND}}), \log n)$ for all Boolean functions $f : \{0,1\}^n \to \{0,1\}$, nearly resolving the log-rank conjecture for AND functions.

While we have a quadratically worse dependence in the RHS of Equation (2) as compared to the above-mentioned bound for symmetric functions due to Buhrman and de Wolf, our upper bound is on a complexity measure that can be exponentially larger than communication complexity in general (Theorem 1.7).

Buhrman and de Wolf showed a lower bound on $\log \operatorname{rank}(M_{f \circ \mathsf{AND}})$ for symmetric functions f. An upper bound on $\mathsf{NAADT}(f)$ implicitly follows from prior work on group testing [14], but we provide a self-contained probabilistic proof for completeness. Combining these two results yields Equation (2), and hence Theorem 1.8.

Suitable analogues of Theorem 1.7 and Theorem 1.8 can be easily seen to hold when the inner gadget is OR instead of AND. In this case, the relevant decision tree model is non-adaptive OR decision trees. Interestingly, these decision trees are studied in the seemingly different context of *non-adaptive group testing algorithms.* Non-adaptive group testing is an active area of research (see, for, example, [12] and the references therein), and has additionally gained significant interest of late in view of the ongoing pandemic (see, for example, [50]).

Our final result regarding the AND gadget deals with the relationship between one-way communication complexity and rank of the underlying communication matrix. It is easy to show that for functions $F : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$,

$$\log \operatorname{rank}(M_F) \leq \mathsf{D}^{\rightarrow}_{\mathrm{cc}}(F) \leq \operatorname{rank}(M_F), \tag{3}$$

where M_F denotes the communication matrix of F and is defined by $M_F(x,y) = F(x,y)$, and rank(·) denotes real rank. The first bound can be seen to be tight for functions with maximal rank, for example the Equality function. The second inequality is tight, for example, for the Addressing function on $(\log n + n)$ input bits (see Definition A.1) where Alice receives n target bits and Bob receives $\log n$ addressing bits. Sanyal [41] showed that the upper bound can be improved for functions of the form $F = f \circ \mathsf{XOR}$. More precisely they showed that for all Boolean functions $f : \{0,1\}^n \to \{0,1\}$,

$$\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{XOR}) \leq O\left(\sqrt{\operatorname{rank}(M_{f\circ\mathsf{XOR}})} \log \operatorname{rank}(M_{f\circ\mathsf{XOR}})\right), \tag{4}$$

and moreover this bound is tight up to the logarithmic factor on the RHS, when f is the Addressing function. We show that the same bound does not hold when the XOR gadget is replaced by AND. We show that (see [33, Corollary A.5]) when f is the Addressing function, then

$$\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND}) \geq \operatorname{rank}(M_{f\circ\mathsf{AND}})^{\log_3 2} \approx \operatorname{rank}(M_{f\circ\mathsf{AND}})^{0.63}, \tag{5}$$

Thus it is plausible that the upper bound in terms of rank from Equation (3) might be tight for some function of the form $f \circ \mathsf{AND}$. We show that this is not the case.

▶ **Theorem 1.9.** *Let $f : \{0,1\}^n : \{0,1\}$ be a Boolean function. Then,*

$$\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND}) \leq (\operatorname{rank}(M_{f\circ\mathsf{AND}}))(1 - \Omega(1)).$$

We show that $\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND})$ is equal to the logarithm of a measure that we define in this work: the *Möbius pattern complexity* of f, which is the total number of distinct evaluations of the monomials in the Möbius expansion of f (see Section 2 for a formal definition of Möbius expansion).

▶ **Definition 1.10** (Möbius pattern complexity). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $f = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S$ be its Möbius expansion. For an input $x \in \{0,1\}^n$, define the* pattern *of x to be $(\mathsf{AND}_S(x))_{S\in\mathcal{S}_f}$. Define the* Möbius pattern complexity *of f, denoted $\mathsf{Pat}^{\mathsf{M}}(f)$, by* $\mathsf{Pat}^{\mathsf{M}}(f) := \left|\left\{P \in \{0,1\}^{\mathcal{S}_f} : P = (\mathsf{AND}_S(x))_{S\in\mathcal{S}_f} \text{ for some } x \in \{0,1\}^n\right\}\right|$.

When clear from context, we refer to the Möbius pattern complexity of f just as the pattern complexity of f.

All of our results involving bounds for $\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND})$ use the above-mentioned equivalence between it and $\log(\mathsf{Pat}^{\mathsf{M}}(f))$ (see Claim 4.1). We unravel interesting mathematical structure in the Möbius supports of Boolean functions, and use them to bound their pattern complexity. We hope that pattern complexity will prove useful in future research.

Organization

We introduce the necessary preliminaries in Section 2. In Section 3 we prove our results regarding the Inner Product gadget (Theorem 1.1 and Theorem 1.3). In Section 4 we prove our results regarding the AND gadget (Theorem 1.7 and Theorem 1.8). We provide remaining preliminaries and missing proofs from the main text in the remaining appendices. Due to space constraints, some proofs are deferred to the full version of our paper [33].

2 Preliminaries

All logarithms in this paper are taken base 2. We use the notation $[n]$ to denote the set $\{1, \ldots, n\}$. We often identify subsets of $[n]$ with their corresponding characteristic vectors in $\{0,1\}^n$. The view we take will be clear from context. Let $\mathsf{S} \subseteq \{0,1\}^n$ be an arbitrary subset of the Boolean hypercube, and let $f : \mathsf{S} \to \{0,1\}$ be a partial Boolean function. If $\mathsf{S} = \{0,1\}^n$, then f is said to be a total Boolean function. When not explicitly mentioned otherwise, we assume Boolean functions to be total.

▶ **Definition 2.1** (Binary entropy). *For $p \in (0,1)$, the binary entropy of p, $\mathbb{H}_{\mathsf{bin}}(p)$, is defined to be the Shannon entropy of a random variable taking two distinct values with probabilities p and $1-p$.*

$$\mathbb{H}_{\mathsf{bin}}(p) := p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}.$$

We now define the Inner Product Modulo 2 function on $2b$ input bits, denoted IP (we drop the dependence of IP on b for convenience; the value of b will be clear from context).

▶ **Definition 2.2** (Inner Product Modulo 2). *For an integer $b > 0$, define the* Inner Product Modulo 2 *function, denoted $\mathsf{IP} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ by $\mathsf{IP}(x_1, \ldots, x_b, y_1, \ldots, y_b) = \oplus_{i \in [b]}(\mathsf{AND}(x_i, y_i))$.*

If f is a partial function, so is $f \circ \mathsf{IP}$.

▶ **Definition 2.3** (Odd-Max-Bit). *Define the* Odd-Max-Bit *function,*[1] *denoted $\mathsf{OMB}_n : \{0,1\}^n \to \{0,1\}$, by $\mathsf{OMB}_n(x) = 1$ if $\max\{i \in [n] : x_i = 0\}$ is odd, and $\mathsf{OMB}_n(x) = 0$ otherwise. Define $\mathsf{OMB}_n(1^n) = 0$.*

Möbius Expansion of Boolean Functions

Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ has a unique expansion as $f = \sum_{S \subseteq [n]} \widetilde{f}(S) \mathsf{AND}_S$, where AND_S denotes the AND of the input variables in S and each $\widetilde{f}(S)$ is a real number. We refer to the functions AND_S as *monomials*, the expansion as the *Möbius expansion* of f, and the real coefficients $\widetilde{f}(S)$ as the Möbius coefficients of f. It is known [3] that the Möbius coefficients can be expressed as $\widetilde{f}(S) = \sum_{X \subseteq S} (-1)^{|S \setminus X|} f(X)$. Define the *Möbius support* of f, denoted $\mathcal{S}_f$, to be the set $\mathcal{S}_f := \left\{ S \subseteq [n] : \widetilde{f}(S) \neq 0 \right\}$. Define the *Möbius sparsity* of f, denoted $\mathrm{spar}(f)$, to be $\mathrm{spar}(f) := |\mathcal{S}_f|$.

Decision Trees and Their Variants

For a partial Boolean function $f : \mathsf{S} \to \{0,1\}$, the deterministic non-adaptive query complexity (alternatively the non-adaptive decision tree complexity) $\mathsf{D}^{\rightarrow}_{\mathrm{dt}}(f)$ is the minimum integer k such that the following is true: there exist k indices $i_1, \ldots, i_k \in [n]$, such that for every Boolean assignment $a_{i_1}, \ldots, a_{i_k}$ to the input variables $x_{i_1}, \ldots, x_{i_k}$, f is constant on $\mathsf{S} \cap \{x \in \{0,1\}^n \mid \forall j = 1, \ldots, k, x_{i_j} = a_{i_j}\}$. Equivalently $\mathsf{D}^{\rightarrow}_{\mathrm{dt}}(f)$ is the minimum number of variables such that f can be expressed as a function of these variables. It is easy to see that if f is a total function that depends on all input variables, then $\mathsf{D}^{\rightarrow}_{\mathrm{dt}}(f) = n$.

[1] In the literature, OMB_n is typically defined with a 1 in the max instead of 0. That function behaves very differently from our OMB_n. For example, it is known that even the *weakly unbounded-error* communication complexity of $\mathsf{OMB}_n \circ \mathsf{AND}$ (under the standard definition of OMB_n) is polynomially large in n [7]. In contrast, it is easy to show that even the deterministic one-way communication complexity of $\mathsf{OMB}_n \circ \mathsf{AND}$ equals $\lceil \log(n+1) \rceil$ with our definition (see Theorem 4.8).

Define the *non-adaptive parity decision tree complexity* of $f : \{0,1\}^n \to \{0,1\}$, denoted by $\mathsf{NAPDT}(f)$, to be the minimum number of parities such that f can be expressed as a function of these parities. Define the *non-adaptive AND decision tree complexity* of $f : \{0,1\}^n \to \{0,1\}$, denoted by $\mathsf{NAADT}(f)$, to be the minimum number of monomials such that f can be expressed as a function of these monomials. Any set of monomials $\mathcal{S}$ whose evaluations determine f is called an *NAADT basis* for f. We also require the natural randomized and quantum analogues of non-adaptive AND decision tree complexity, denoted $\mathsf{RNAADT}(\cdot)$ and $\mathsf{QNAADT}(\cdot)$, respectively. Formal definitions of these measures can be found in Appendix A. We first note some simple observations about the non-adaptive AND decision tree complexity of Boolean functions.

▷ Claim 2.4. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\mathcal{S} = \{S_1, \dots, S_k\}$ be a NAADT basis for f. Then, every monomial in the Möbius support of f equals $\prod_{i \in T} \mathsf{AND}_{S_i}$, for some $T \subseteq [k]$.

Proof. Since $\mathcal{S}$ is an NAADT basis for f, the values of $\{\mathsf{AND}_{S_i} : i \in [k]\}$ determine the value of f. That is, we can express f as

$$f = \sum_{T \subseteq [k]} b_T \prod_{i \in T} \mathsf{AND}_{S_i} \prod_{j \notin T} (1 - \mathsf{AND}_{S_j}),$$

for some values of $b_T \in \{0,1\}$. Expanding this expression only yields monomials that are products of AND_{S_i}'s from $\mathcal{S}$. The claim now follows since the Möbius expansion of a Boolean function is unique. ◁

▷ Claim 2.5. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with Möbius sparsity r. Then $\log r \leq \mathsf{NAADT}(f) \leq r$.

Proof. The upper bound $\mathsf{NAADT}(f) \leq r$ follows from the fact that knowing the values of all ANDs in the Möbius support of f immediately yields the value of f by plugging these values in the Möbius expansion of f. That is, the Möbius support of f acts as an NAADT basis for f.

For the lower bound, let $\mathsf{NAADT}(f) = k$, and let $\mathcal{S} = \{S_1, \dots, S_k\}$ be an NAADT basis for f. Claim 2.4 implies that every monomial in the Möbius expansion of f is a product of some of these AND_{S_i}'s. Thus, the Möbius sparsity of f is at most 2^k, yielding the required lower bound. ◁

Every Boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be uniquely written as $f = \sum_{S \subseteq [n]} \widehat{f}(S)(-1)^{\oplus_{j \in S} x_j}$. This representation is called the *Fourier expansion* of f and the real values $\widehat{f}(S)$ are called the Fourier coefficients of f. The Fourier sparsity of f is defined to be number of non-zero Fourier coefficients of f. Sanyal [41] showed the following relationship between non-adaptive parity decision complexity of a Boolean function and its Fourier sparsity.

▶ **Theorem 2.6** ([41]). *Let $f : \{0,1\}^n \to \{-1,1\}$ be a Boolean function with Fourier sparsity r. Then $\mathsf{NAPDT}(f) = O(\sqrt{r} \log r)$.*

This theorem is tight up to the logarithmic factor, witnessed by the Addressing function.

Communication Complexity

The standard model of two-party communication complexity was introduced by Yao [49]. In this model, there are two parties, say Alice and Bob, each with inputs $x, y \in \{0,1\}^n$. They wish to jointly compute a function $F(x, y)$ of their inputs for some function $F : \mathcal{U} \to \{0,1\}$ that is known to them, where $\mathcal{U}$ is a subset of $\{0,1\}^n \times \{0,1\}^n$. They use a communication protocol agreed upon in advance. The cost of the protocol is the number of bits exchanged in the worst case (over all inputs). Alice and Bob are required to output the correct answer for all inputs $(x, y) \in \mathcal{U}$. The communication complexity of F is the best cost of a protocol that computes F, and we denote it by $\mathsf{D}_{\mathrm{cc}}(F)$. See, for example, [29], for an introduction to communication complexity.

In a deterministic one-way communication protocol, Alice sends a message $m(x)$ to Bob. Then Bob outputs a bit depending on $m(x)$ and y. The complexity of the protocol is the maximum number of bits a message contains for any input x to Alice. In a randomized one-way protocol, the parties share some common random bits $\mathcal{R}$. Alice's message is a function of x and $\mathcal{R}$. Bob's output is a function of $m(x), y$ and $\mathcal{R}$. The protocol Π is said to compute F with error $\varepsilon \in (0, 1/2)$ if for every $(x, y) \in \mathcal{U}$, the probability over $\mathcal{R}$ of the event that Bob's output equals $F(x, y)$ is at least $1 - \varepsilon$. The cost of the protocol is the maximum number of bits contained in Alice's message for any x and $\mathcal{R}$. In the one-way quantum model, Alice sends Bob a quantum message, after which Bob performs a projective measurement and outputs the measurement outcome. Depending on the model of interest, Alice and Bob may or may not share an arbitrary input-independent entangled state for free. We refer the reader to [46] for an introduction to quantum communication complexity. As in the randomized setting, a protocol Π computes F with error ε if $\Pr[\Pi(x, y) \neq f(x, y)] \leq \varepsilon$ for all $(x, y) \in \mathcal{U}$.

The deterministic (ε-error randomized, ε-error quantum, ε-error quantum with entanglement, respectively) one-way communication complexity of F, denoted by $\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(\cdot)$ ($\mathsf{R}^{\rightarrow}_{\mathrm{cc},\varepsilon}(\cdot)$, $\mathsf{Q}^{\rightarrow}_{\mathrm{cc},\varepsilon}(\cdot)$, $\mathsf{Q}^{*,\rightarrow}_{\mathrm{cc},\varepsilon}(\cdot)$, respectively), is the minimum cost of any deterministic (ε-error randomized, ε-error quantum, ε-error quantum with entanglement, respectively) one-way communication protocol for F.

Total functions F whose domain is $\{0,1\}^n \times \{0,1\}^n$ induce a communication matrix M_F whose rows and columns are indexed by strings in $\{0,1\}^n$, and the (x, y)'th entry equals $F(x, y)$. It is known that

$$\log \operatorname{rank}(M_F) \leq \mathsf{D}_{\mathrm{cc}}(F) \leq O(\sqrt{\operatorname{rank}(M_F)} \log \operatorname{rank}(M_F)), \tag{6}$$

where $\operatorname{rank}(\cdot)$ denotes real rank. The first inequality is well known (see, for instance [29]), and the second inequality was shown by Lovett [32]. One of the most famous conjectures in communication complexity is the log-rank conjecture, due to Lovász and Saks [31], that proposes that the communication complexity of any Boolean function is polylogarithmic in its rank, i.e. the first inequality in Equation (6) is always tight up to a polynomial dependence.

Buhrman and de Wolf [8] observed that the Möbius sparsity of a Boolean function f equals the rank of the communication matrix of $f \circ \mathsf{AND}$. That is, for all Boolean functions $f : \{0,1\}^n \to \{0,1\}$,

$$\operatorname{spar}(f) = \operatorname{rank}(M_{f \circ \mathsf{AND}}). \tag{7}$$

In view of the first inequality in Equation (6), this yields

$$\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND}) \geq \log(\operatorname{spar}(f)). \tag{8}$$

We require the definition of the Vapnik-Chervonenkis (VC) dimension [45].

▶ **Definition 2.7** (VC-dimension). *Consider a function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. A subset of columns C of M_F is said to be* shattered *if all of the $2^{|C|}$ patterns of 0's and 1's are attained by some row of M_F when restricted to the columns C. The* VC-dimension *of a function $F : \{0,1\}^n \times \{0,1\}^n$, denoted $\mathsf{VC}(F)$, is the maximum size of a shattered subset of columns of M_F.*

Klauck [26] showed that the one-way quantum communication complexity of a function F is bounded below by the VC-dimension of F.

▶ **Theorem 2.8** ([26, Theorem 3]). *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then, $\mathsf{Q}^{\to}_{\mathsf{cc},\varepsilon}(F) \geq (1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon))\mathsf{VC}(F)$ and $\mathsf{Q}^{*,\to}_{\mathsf{cc},\varepsilon}(F) \geq (1 - \mathbb{H}_{\mathsf{bin}}(\varepsilon))\mathsf{VC}(F)/2$.*

3 Composition with Inner Product

In this section we prove Theorem 1.1 and Theorem 1.3, which are our results regarding the quantum and deterministic one-way communication complexities, respectively, of functions composed with a small Inner Product gadget.

Quantum Complexity

Proof of Theorem 1.1. By Theorem 2.8, it suffices to show that $\mathsf{VC}(f \circ \mathsf{IP}) \geq n(b-1)$. Since f is a function that depends on all its input variables, the following holds. For each index $i \in [n]$, there exist inputs $z^{(i,0)} = z^{(i)}_1, \ldots, z^{(i)}_{i-1}, 0, z^{(i)}_{i+1}, \ldots, z^{(i)}_n$ and $z^{(i,1)} = z^{(i)}_1, \ldots, z^{(i)}_{i-1}, 1, z^{(i)}_{i+1}, \ldots, z^{(i)}_n$ such that $f(z^{(i,0)}) = v_i$ and $f(z^{(i,1)}) = 1 - v_i$. That is, $z^{(i,0)}$ and $z^{(i,1)}$ have different function values, but differ only on the i'th bit.

For each $i \in [n]$ and $j \in \{2, 3, \ldots, b\}$, define a string $y^{(i,j)} \in \{0,1\}^{nb}$ as follows. For all $k \in [n]$ and $\ell \in [b]$,

$$y^{(i,j)}_{k,\ell} = \begin{cases} z^{(i)}_k & \text{if } k \neq i \text{ and } \ell = 1 \\ 1 & \text{if } k = i \text{ and } \ell = j \\ 0 & \text{otherwise.} \end{cases}$$

That is, for $k \neq i$, the k'th block of $y^{(i,j)}$ is $(z^{(i)}_k, 0^{b-1})$, and the i'th block of $y^{(i,j)}$ is $(0^{j-1}, 1, 0^{b-j})$. Consider the set of $n(b-1)$-many columns of $M_{f \circ \mathsf{IP}}$, one for each $y^{(i,j)}$. We now show that this set of columns is shattered. Consider an arbitrary string $c = c_{1,2}, \ldots, c_{1,b}, \ldots, c_{n,2}, \ldots, c_{n,b} \in \{0,1\}^{n(b-1)}$. We now show the existence of a row that yields this string on restriction to the columns described above. Define a string $x \in \{0,1\}^{nb}$ as follows. For all $i \in [n]$ and $j \in [b]$, $x_{i,1} = 1$ and

$$x_{i,j} = \begin{cases} c_{i,j} & \text{if } v_i = 0 \\ 1 - c_{i,j} & \text{if } v_i = 1. \end{cases}$$

That is, the first element of each block of x is 1, and the remaining part of any block, say the i'th block, equals either the string $c_{i,2}, \ldots, c_{i,b}$ or its bitwise negation, depending on the value of v_i.

To complete the proof, we claim that the row of $M_{f \circ \mathsf{IP}}$ corresponding to this string x equals the string c when restricted to the columns $\{y^{(i,j)}\}_{i \in [n], j \in \{2,3,\ldots,b\}}$. To see this, fix $i \in [n]$ and $j \in \{2, 3, \ldots, b\}$ and consider $M_{f \circ \mathsf{IP}}(x, y^{(i,j)})$. Next, for each $k \in [n]$ with $k \neq i$, the inner product of the k'th block of x with the k'th block of y equals $z^{(i)}_k$, since $x_{k,1} = 1$

and the first element of the k'th block of $y^{(i,j)}$ equals $z_k^{(i)}$, and all other elements of the block are 0 by definition. In the i'th block of $y^{(i,j)}$, only the j'th element is non-zero, and equals 1 by definition. Moreover, $x_{i,j} = c_{i,j}$ if $v_i = 0$, and equals $1 - c_{i,j}$ otherwise. Hence, the inner products of the i'th blocks of x and $y^{(i,j)}$ equals $c_{i,j}$ if $v_i = 0$, and equals $1 - c_{i,j}$ otherwise. Thus, the string obtained on taking the block-wise inner product of x and $y^{(i,j)}$ equals $z_1^{(i)}, \ldots, z_{i-1}^{(i)}, c_{i,j}, z_{i+1}^{(i)}, \ldots, z_n^{(i)}$ if $v_i = 0$ and $z_1^{(i)}, \ldots, z_{i-1}^{(i)}, 1 - c_{i,j}, z_{i+1}^{(i)}, \ldots, z_n^{(i)}$ if $v_i = 1$. By our definitions of $z^{(i,0)}, z^{(i,1)}$ and v_i for each $i \in [n]$, it follows that the value of f when applied to either of these inputs equals $c_{i,j}$. This concludes the proof. ◀

Deterministic Complexity

We now prove Theorem 1.3, which gives a lower bound on the deterministic one-way communication complexity of $f \circ \mathsf{IP}$ for partial functions f. A crucial ingredient of our proof is Theorem 1.4. Now we proceed to the proof of Theorem 1.3.

Proof of Theorem 1.3. Let $q := 2^b - 1$ and let Π be an optimal one-way deterministic protocol for $f \circ \mathsf{IP}$ of complexity $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{IP}) =: c \log q$. The theorem is trivially true if $c \geq n/30$ since $\mathsf{D}_{\mathrm{dt}}^{\rightarrow}(f) \leq n$. In the remainder of the proof we assume that $c < n/30$. Π induces a partition of $\{0,1\}^{nb}$ into at most q^c parts; each part corresponds to a distinct message. There are $(2^b - 1)^n = q^n$ inputs $(x_1, \ldots, x_n)$ to Alice such that for each i, $x_i \neq 0^b$. Let $\mathcal{Z}$ be the set of those inputs. Identify $\mathcal{Z}$ with $[q]^n$. By the pigeon-hole principle there exists one part P in the partition induced by Π that contains at least q^{n-c} strings in $\mathcal{Z}$. We now invoke Theorem 1.4 with d set to $10c$. This is applicable since $d \leq n/3$ and the assumption $b \geq 2$ implies that $q \geq 3$. Theorem 1.4 implies that there are two strings $x^{(1)} = (x_1^{(1)}, \ldots, x_n^{(1)}), x^{(2)} = (x_1^{(2)}, \ldots, x_n^{(2)}) \in \mathsf{P} \cap \mathcal{Z}$ such that $|\{i \in [n] \mid x_i^{(1)} = x_i^{(2)}\}| < 10c$. Let $\mathcal{I} := \{i \in [n] \mid x_i^{(1)} = x_i^{(2)}\}$. Let $z = (z_1, \ldots, z_n)$ denote a generic input to f. We claim that for each Boolean assignment $(a_i)_{i \in \mathcal{I}}$ to the variables in $\mathcal{I}$, f is constant on $\mathsf{S} \cap \{z : \forall i \in \mathcal{I}, z_i = a_i\}$. This will prove the theorem, since querying the variables $\{z_i \mid i \in \mathcal{I}\}$ determines f; thus $\mathsf{D}_{\mathrm{dt}}^{\rightarrow}(f) \leq |\mathcal{I}| < 10c$. Towards a contradiction, assume that there exist $z^{(1)}, z^{(2)} \in \mathsf{S} \cap \{z : \forall i \in \mathcal{I}, z_i = a_i\}$ such that $f(z^{(1)}) \neq f(z^{(2)})$. We will construct a string $y = (y_1, \ldots, y_n) \in \{0,1\}^{nb}$ in the following way:
$i \in \mathcal{I}$: Choose y_i such that $\mathsf{IP}(y_i, x_i^{(1)}) = \mathsf{IP}(y_i, x_i^{(2)}) = a_i$.
$i \notin \mathcal{I}$: Choose y_i such that $\mathsf{IP}(y_i, x_i^{(1)}) = z_i^{(1)}$ and $\mathsf{IP}(y_i, x_i^{(2)}) = z_i^{(2)}$.
Note that we can always choose a y as above since for each $i \in [n]$, $x_i^{(1)}, x_i^{(2)} \neq 0^b$, and for each $i \notin \mathcal{I}$, $x_i^{(1)} \neq x_i^{(2)}$. By the above construction, $f \circ \mathsf{IP}(x^{(1)}, y) = f(z^{(1)})$ and $f \circ \mathsf{IP}(x^{(2)}, y) = f(z^{(2)})$. Since by assumption $f(z^{(1)}) \neq f(z^{(2)})$, we have that $f \circ \mathsf{IP}(x^{(1)}, y) \neq f \circ \mathsf{IP}(x^{(2)}, y)$. But since Alice sends the same message on inputs $x^{(1)}$ and $x^{(2)}$, Π produces the same output on $(x^{(1)}, y)$ and $(x^{(2)}, y)$. This contradicts the correctness of Π. ◀

▶ Remark 3.1. It can be seen that the proof of Theorem 1.3 also works when the inner gadget $g : \{0,1\}^{b_1} \times \{0,1\}^{b_2} \to \{0,1\}$ satisfies the following general property: There exists a subset X of $\{0,1\}^{b_1}$ (Alice's input in the gadget) such that:

- $|X| \geq 3$,
- for all $x_1 \neq x_2 \in X$ and all $b_1, b_2 \in \{0,1\}$, there exists $y \in \{0,1\}^{b_2}$ such that $g(x_1, y) = b_1$ and $g(x_2, y) = b_2$.

This is satisfied, for example, for the Addressing function on $\{0,1\}^{\log b + b}$ when $b \geq 4$ (see Definition A.1). For $g = \mathsf{IP}_b$, the set X equals $\{0,1\}^b \setminus \{0^b\}$.

4 Composition with AND

We first investigate the relationship between non-adaptive AND decision tree complexity and Möbius sparsity of Boolean functions. Recall that Claim 2.5 shows that for all Boolean functions $f : \{0,1\}^n \to \{0,1\}$, $\log \mathrm{spar}(f) \leq \mathsf{NAADT}(f) \leq \mathrm{spar}(f)$. A natural question to ask is whether both of the bounds are tight, i.e. are there Boolean functions witnessing tightness of each bound? The first bound is trivially tight for any Boolean function with full Möbius sparsity, for example, the NOR function: querying all the input bits (which is querying n many ANDs) immediately yields the value of the function, and its Möbius sparsity can be shown to be 2^n. One might expect that the upper bound is not tight in view of Theorem 2.6. The Addressing function witnesses tightness of the quadratic gap in Theorem 2.6. This gives rise to the natural question of whether an analogous bound holds true in the Möbius-world: is it true for all Boolean functions f that $\mathsf{NAADT}(f) = \widetilde{O}(\sqrt{\mathrm{spar}(f)})$? Interestingly we show (see [33, Appendix A]) that the Addressing function already gives a negative answer to this question. In Claim 4.6 we observe that the function OMB_n witnesses tightness of the second inequality in Claim 2.5, that is, $\mathsf{NAADT}(\mathsf{OMB}_n) = \mathrm{spar}(f)$ for even n (and $\mathsf{NAADT}(\mathsf{OMB}_n) = \mathrm{spar}(f) - 1$ for odd n). We then use this same function to prove Theorem 1.7, which gives a maximal separation between $\mathsf{QNAADT}(f)$ and $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}_2)$. Finally, we prove Theorem 1.8, which says that $\mathsf{NAADT}(f)$ is at most quadratically large in $\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND})$ for symmetric f.

Pattern Complexity and One-Way Communication Complexity

In this section we observe that the logarithm of the pattern complexity, $\mathsf{Pat}^{\mathsf{M}}(f)$, of a Boolean function f equals the deterministic one-way communication complexity of $f \circ \mathsf{AND}$. We also give bounds on $\mathsf{NAADT}(f)$ in terms of $\mathsf{Pat}^{\mathsf{M}}(f)$. As a consequence we also show that $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \geq \log(\mathsf{NAADT}(f))$.

▷ Claim 4.1. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) = \lceil \log(\mathsf{Pat}^{\mathsf{M}}(f)) \rceil$.

Proof. Write the Möbius expansion of f as

$$f = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S) \mathsf{AND}_S. \tag{9}$$

Say $\mathsf{Pat}^{\mathsf{M}}(f) = k$. We first show that $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \leq \lceil \log k \rceil$ by exhibiting a one-way protocol of cost $\lceil \log k \rceil$. Alice computes the pattern of x and sends Bob the pattern using $\lceil \log k \rceil$ bits of communication. Bob now knows the values of $\{\mathsf{AND}_S(x) : S \in \mathcal{S}_f\}$. Since Bob can compute $\{\mathsf{AND}_S(y) : S \in \mathcal{S}_f\}$ without any communication, he can now compute the value of $f \circ \mathsf{AND}(x, y)$ using the formula

$$(f \circ \mathsf{AND})(x, y) = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S) \mathsf{AND}_S(x) \mathsf{AND}_S(y).$$

It remains to show that $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \geq \lceil \log k \rceil$. Let $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) = d$. Thus there are at most 2^d messages that Alice can send Bob. We show that any two inputs $x, x' \in \{0,1\}^n$ for which Alice sends the same message have the same pattern, which would prove $2^d \geq k$, and prove the claim since d must be an integer.

Let x, x' be 2 inputs to Alice for which her message to Bob is m. We have

$$(f \circ \mathsf{AND})(x, y) = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S(x)\mathsf{AND}_S(y)$$
$$(f \circ \mathsf{AND})(x', y) = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S(x')\mathsf{AND}_S(y)$$

Since m and y completely determine the value of the function, we must have

$$\sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S(x)\mathsf{AND}_S(y) = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S(x')\mathsf{AND}_S(y) \qquad \text{for all } y \in \{0,1\}^n.$$

Define the functions $g_x, g_{x'} : \{0,1\}^n \to \{0,1\}$ by

$$g_x(y) = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S(x)\mathsf{AND}_S(y)$$
$$g_{x'}(y) = \sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S(x')\mathsf{AND}_S(y).$$

Thus by uniqueness of the Möbius expansion of Boolean functions, $g_x = g_{x'}$ as functions of y. This implies $\widetilde{g_x}(S) = \widetilde{g_{x'}}(S)$ for all $S \in \mathcal{S}_f$. Since $\widetilde{g_x}(S) = \widetilde{f}(S)\mathsf{AND}_S(x)$ and $\widetilde{g_{x'}}(S) = \widetilde{f}(S)\mathsf{AND}_S(x')$ for all $S \in \mathcal{S}_f$,

$$\mathsf{AND}_S(x) = \mathsf{AND}_S(x') \qquad \text{for all } S \in \mathcal{S}_f,$$

This shows that the pattern induced by x and the pattern induced by x' are the same, concluding the proof. ◁

Next we show that the pattern complexity of f is bounded below by the Möbius sparsity of f.

▷ Claim 4.2. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\mathsf{Pat}^{\mathsf{M}}(f) \geq \mathrm{spar}(f)$.

Proof. Recall that $\mathcal{S}_f$ denotes the Möbius support of f. For each $S \in \mathcal{S}_f$, define the input x^S to be the n-bit characteristic vector of the set S. We now show that each of these inputs induces a different pattern for f. Let $S_1 \neq S_2 \in \mathcal{S}_f$, with $|S_1| \geq |S_2|$. Since they are different sets, there must be an index $j \in S_1$ such that $j \notin S_2$. Note that $\mathsf{AND}_{S_1}(x^{S_1}) = 1$. On the other hand $x_j^{S_2} = 0$ implies $\mathsf{AND}_{S_1}(x^{S_2}) = 0$. Hence x^{S_1} and x^{S_2} induce different patterns. Since $\mathrm{spar}(f) = |\mathcal{S}_f|$, this completes the proof. ◁

From Claim 2.5 we know that $\mathrm{spar}(f) \geq \mathsf{NAADT}(f)$ and from Claim 4.1 we know that $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) = \lceil \log(\mathsf{Pat}^{\mathsf{M}}(f)) \rceil$. Along with Claim 4.2, these imply the following claim.

▷ Claim 4.3. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\lceil \log(\mathsf{NAADT}(f)) \rceil \leq \mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND}) \leq \mathsf{NAADT}(f)$.

Proof. For the upper bound on $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(f \circ \mathsf{AND})$, let $\mathcal{S} = \{S_1, \ldots, S_k\}$ be an NAADT basis for f. By Claim 2.4, every monomial in the Möbius support of f is a product of some of these AND_{S_i}'s. Since there are at most 2^k possible values for $\{\mathsf{AND}_{S_i}(x) : i \in [k]\}$ and since these completely determine the pattern of x for any given $x \in \{0,1\}^n$, we have

$$\mathsf{Pat}^{\mathsf{M}}(f) \leq 2^{\mathsf{NAADT}(f)},$$

which proves the required upper bound in view of Claim 4.1.

For the lower bound, we have

$$\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND}) = \lceil \log(\mathsf{Pat}^{\mathsf{M}}(f)) \rceil \geq \lceil \log(\mathrm{spar}(f)) \rceil \geq \lceil \log(\mathsf{NAADT}(f)) \rceil,$$

where the equality follows from Claim 4.1, the first inequality follows from Claim 4.2 and the last inequality follows from Claim 2.5. ◁

The pattern complexity of f is trivially at most $2^{\mathrm{spar}(f)}$ since each pattern is a $\mathrm{spar}(f)$-bit string. Interestingly we show that there is no function for which this bound is tight.

▷ Claim 4.4. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\mathsf{Pat}^{\mathsf{M}}(f) \leq 2^{(1-\Omega(1))\mathrm{spar}(f)}$.

We prove Claim 4.4 in Appendix B. Its proof proceeds by analyzing the identity $f^2 = f$ and using it to deduce “dependencies” between monomials in the Möbius support of f. The analogous relation in the Fourier-world has been nearly determined by Sanyal [41]; their main result (Theorem 2.6) essentially shows that the Fourier analog of pattern complexity of a Boolean function is at most exponential in the square root of its Fourier sparsity. This is a stronger bound than that in Claim 4.4, but the same bound cannot hold in the Möbius-world since the Addressing function witnesses $\mathsf{Pat}^{\mathsf{M}}(\mathsf{ADDR}_n) \geq 2^{\mathrm{spar}(\mathsf{ADDR}_n)^{\log_3 2}}$ (see [33, Appendix A]). Nevertheless we conjecture that a stronger bound than that of Claim 4.4 is possible.

▶ **Conjecture 4.5.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then* $\mathsf{Pat}^{\mathsf{M}}(f) \leq 2^{\left(\mathrm{spar}(f)^{1-\Omega(1)}\right)}$.

Conjecture 4.5 would strengthen Theorem 1.9, showing that $\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND}) = \mathrm{rank}(M_{f \circ \mathsf{AND}})^{1-\Omega(1)}$.

Proof of Theorem 1.9. We have

$$\mathsf{D}^{\rightarrow}_{\mathrm{cc}}(f \circ \mathsf{AND}) = \lceil \log(\mathsf{Pat}^{\mathsf{M}}(f)) \rceil \leq (1-\Omega(1))\mathrm{spar}(f) \leq (1-\Omega(1))\mathrm{rank}(M_{f \circ \mathsf{AND}}),$$

where the equality holds by Claim 4.1, the first inequality follows from Claim 4.4 and the last inequality holds by Equation (7). ◀

Our results regarding the one-way communication complexity of $f \circ \mathsf{AND}$ use the Booleanness of f to bring out mathematical insights about the dependencies of monomials in the Möbius support of f. These dependencies enable us to establish interesting bounds on the pattern complexity of f. We hope that pattern complexity will find more use in future research.

Deterministic AND Complexity

We prove in this section that the non-adaptive AND decision tree complexity of OMB_n is maximal whereas the one-way communication complexity of $\mathsf{OMB}_n \circ \mathsf{AND}$ is small.

▷ Claim 4.6. Let n be a positive integer. Then $\mathsf{NAADT}(\mathsf{OMB}_n) = n$. Moreover, $\mathrm{spar}(\mathsf{OMB}_n) = n$ if n is even, and $\mathrm{spar}(\mathsf{OMB}_n) = n + 1$ if n is odd.

Proof of Claim 4.6. Write the polynomial representation of OMB_n as $\mathsf{OMB}_n(x) =$

$$(1 - x_n) \cdot 0 + x_n(1 - x_{n-1}) \cdot 1 + x_n x_{n-1} \mathsf{OMB}_{n-2}(x_1, \ldots, x_{n-2}) \quad \text{if } n \text{ is even, or} \tag{10}$$

$$(1 - x_n) \cdot 1 + x_n(1 - x_{n-1}) \cdot 0 + x_n x_{n-1} \mathsf{OMB}_{n-2}(x_1, \ldots, x_{n-2}) \quad \text{if } n \text{ is odd.} \tag{11}$$

The Möbius support of OMB_n equals $\{\{j,\dots,n\} : j \le n\} \cup \{\emptyset\}$ if n is odd, and $\{\{j,\dots,n\} : j \le n\}$ if n is even. Thus $\mathrm{spar}(\mathsf{OMB}_n) = n+1$ if n is odd, and equals n if n is even.

We now show that the $\mathsf{NAADT}(\mathsf{OMB}_n) = n$. Let $\mathcal{S}$ denote a NAADT basis for OMB_n. By Claim 2.4, any monomial in the Möbius expansion of OMB_n can be expressed as a product of some ANDs from $\mathcal{S}$. Thus, $\{n\}$ must participate in $\mathcal{S}$ since it appears in its Möbius support. Next, since $\{n-1, n\}$ appears in the support as well, either $\{n-1, n\}$ or $\{n-1\}$ must appear in $\mathcal{S}$. Continuing iteratively, we conclude that for all $i \in [n]$, there must exist a set in $\mathcal{S}$ that contains i, but does not contain any j for $j < i$. This implies that $|\mathcal{S}| \ge n$. Equality holds since $\mathsf{NAADT}(f) \le n$ for any Boolean function $f : \{0,1\}^n \to \{0,1\}$. ◁

Thus OMB_n witnesses that non-adaptive AND decision tree complexity can be as large as sparsity. We remark here that OMB_n admits a simple (adaptive) AND-decision tree that makes $O(\log n)$ AND-queries in the worst case. This uses a binary search using AND-queries to determine the right-most index where a 0 is present. One might expect that a result similar to Claim 1.6 holds when the inner function is AND instead of XOR. That is, it is plausible that the deterministic one-way communication complexity of $f \circ \mathsf{AND}$ equals the non-adaptive AND decision tree complexity of f. We show that this is not true, and exhibit an exponential separation between $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(\mathsf{OMB}_n \circ \mathsf{AND})$ and $\mathsf{NAADT}(\mathsf{OMB}_n)$.

▷ Claim 4.7. Let n be a positive integer. Then $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(\mathsf{OMB}_n \circ \mathsf{AND}) = \lceil \log(n+1) \rceil$.

Proof. From Equation (10) we have that the Möbius support of OMB_n equals the set $\mathcal{S} = \{\{n\}, \{n-1, n\}, \dots, \{n, n-1, \dots, 1\}\}$ if n is an even integer, and equals the set $\mathcal{S} = \{\emptyset, \{n\}, \{n-1, n\}, \dots, \{n, n-1, \dots, 1\}\}$ if n is an odd integer. It is easy to verify that the only possible Möbius patterns attainable (ignoring the empty set since it always evaluates to 1) are $1^i 0^{n-i}$, for $i \in \{0, 1, \dots, n\}$. Moreover, all of these patterns are attainable: the pattern $1^i 0^{n-i}$ is attained by the input string $0^{n-i} 1^i$. Thus $\mathsf{Pat}^{\mathsf{M}}(\mathsf{OMB}_n) = n+1$. Claim 4.1 implies $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(\mathsf{OMB}_n \circ \mathsf{AND}) = \lceil \log(n+1) \rceil$. ◁

We obtain our main result of this section, which follows from Claim 4.6 and Claim 4.7.

▶ **Theorem 4.8.** *Let n be a positive integer. Then $\mathsf{NAADT}(\mathsf{OMB}_n) = n$ and $\mathsf{D}_{\mathrm{cc}}^{\rightarrow}(\mathsf{OMB}_n \circ \mathsf{AND}) = \lceil \log(n+1) \rceil$.*

Quantum Complexity

We prove that even the quantum non-adaptive AND decision tree complexity of OMB_n is $\Omega(n)$. We refer the reader to Section A for necessary preliminaries of quantum computing. In view of the small one-way communication complexity of $\mathsf{OMB}_n \circ \mathsf{AND}$ from Claim 4.7, Theorem 1.7 then follows.

▶ **Theorem 4.9.** *Let n be a positive integer. Then $\mathsf{QNAADT}(\mathsf{OMB}_n) = \Omega(n)$.*

Before we prove this theorem, we introduce an auxiliary function and state some properties of it that are of use to us.

▶ **Definition 4.10.** *Let n be a positive integer. Define the set $\mathsf{S} \subset \{0,1\}^n$ to be $\mathsf{S} = \left\{x \in \{0,1\}^n : x = 0^i 1^{n-i} \text{ for some } i \in [n]\right\}$. Define the partial function $\mathsf{OMB}'_n : \mathsf{S} \to \{0,1\}$ by $\mathsf{OMB}'_n(x) = \mathsf{OMB}_n(x)$.*

▷ Claim 4.11. Let n be a positive integer. Then $\mathsf{RNAADT}(\mathsf{OMB}'_n) = \mathsf{R}_{\mathrm{dt}}^{\rightarrow}(\mathsf{OMB}'_n)$ and $\mathsf{QNAADT}(\mathsf{OMB}'_n) = \mathsf{Q}_{\mathrm{dt}}^{\rightarrow}(\mathsf{OMB}'_n)$.

We require the following result, which follows implicitly from a result of Montanaro [34].

▶ **Theorem 4.12.** *Let $\mathsf{S} \subseteq \{0,1\}^n$, $I \subseteq [n]$ and $f : \mathsf{S} \to \{0,1\}$ be such that for all $i \in I$ there exists $x \in \mathsf{S}$ such that $f(x \oplus e_i) = 1 - f(x)$. Then $\mathsf{Q}_{\mathrm{dt}}^{\rightarrow}(f) = \Omega(|I|)$.*

We defer the proofs of Claim 4.11 and Theorem 4.12 to the full version of our paper [33, Section 4.3].

Proof of Theorem 4.9. Clearly $\mathsf{QNAADT}(\mathsf{OMB}_n) \geq \mathsf{QNAADT}(\mathsf{OMB}'_n)$. Claim 4.11 implies that $\mathsf{QNAADT}(\mathsf{OMB}'_n) = \mathsf{Q}_{\mathrm{dt}}^{\rightarrow}(\mathsf{OMB}'_n)$. Recall that the domain of OMB'_n equals $S = \left\{x \in \{0,1\}^n : x = 0^i 1^{n-i} \text{ for some } i \in [n]\right\}$. By definition, $\mathsf{OMB}'_n(0^i 1^{n-i}) \neq \mathsf{OMB}'_n(0^{i-1} 1^{n-i+1})$ for all $i \in [n]$. Thus Theorem 4.12 is applicable with $I = [n]$ and $f = \mathsf{OMB}'_n$. Combining the above, we have $\mathsf{QNAADT}(\mathsf{OMB}_n) \geq \mathsf{QNAADT}(\mathsf{OMB}'_n) = \mathsf{Q}_{\mathrm{dt}}^{\rightarrow}(\mathsf{OMB}'_n) = \Omega(n)$. ◀

Proof of Theorem 1.7. It follows from Claim 4.7 and Theorem 4.9. ◀

Symmetric Functions

In this section we show that symmetric functions f admit efficient non-adaptive AND decision trees in terms of the deterministic (even two-way) communication complexity of $f \circ \mathsf{AND}$. We require the following bounds on the Möbius sparsity of symmetric functions, due to Buhrman and de Wolf [8]. For a non-constant symmetric function $f : \{0,1\}^n \to \{0,1\}$, define the following measure which captures the smallest Hamming weight inputs before which f is not a constant: $\mathsf{switch}(f) := \min\{k : f \text{ is a constant on all } x \text{ such that } |x| < n-k\}$.

▷ Claim 4.13 ([8, Lemma 5]). Let n be sufficiently large, let $f : \{0,1\}^n \to \{0,1\}$ be a symmetric Boolean function, and let $k := \mathsf{switch}(f)$. Then $\log \mathrm{spar}(f) \geq \frac{1}{2} \log\left(\sum_{i=n-k}^{n} \binom{n}{i}\right)$.

Upper bounds on the non-adaptive AND decision tree complexity of symmetric functions follow from known results in the non-adaptive group testing literature. To the best of our knowledge, the following upper bounds were first shown (formulated differently) by Dyachkov and Rykov [14]. Also see [12] and the references therein.

▶ **Theorem 4.14.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a symmetric Boolean function with $\mathsf{switch}(f) = k < n/2$. Then $\mathsf{NAADT}(f) = O\left(\log^2 \binom{n}{k}\right)$.*

We give a self-contained proof of Theorem 4.14 in Appendix C for clarity and completeness. We are now ready to prove Theorem 1.8.

Proof of Theorem 1.8. If $\mathsf{switch}(f) \geq n/2$, then Claim 4.13 implies that $\mathrm{spar}(f) = 2^{\Omega(n)}$. Equation (8) implies that $\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND}) = \Omega(n)$. Thus, a trivial NAADT of cost n witnesses $\mathsf{NAADT}(f) = O(\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND}))$ in this case.

Hence, we may assume $\mathsf{switch}(f) = k < n/2$. We have

$$\mathsf{NAADT}(f) = O\left(\log^2 \binom{n}{k}\right) = O(\log^2(\mathrm{spar}(f))) = O(\mathsf{D}_{\mathrm{cc}}(f \circ \mathsf{AND})^2),$$

where the first equality follows from Theorem 4.14, the second from Claim 4.13, and the third from Equation (8). ◀

References

1 Rudolf Ahlswede and Levon H Khachatrian. The diametric theorem in hamming spaces—optimal anticodes. *Advances in Applied mathematics*, 20(4):429–449, 1998.

2 Anurag Anshu, Dmitry Gavinsky, Rahul Jain, Srijita Kundu, Troy Lee, Priyanka Mukhopadhyay, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 10:1–10:13, 2017.

3 Richard Beigel. The polynomial method in circuit complexity. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 82–95, 1993. doi:10.1109/SCT.1993.336538.

4 Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions: Extended abstract. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 240–246. IEEE, 2020.

5 Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 60:1–60:14, 2016.

6 Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. doi:10.1137/S0097539796300921.

7 Harry Buhrman, Nikolai K. Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *22nd Annual IEEE Conference on Computational Complexity (CCC)*, pages 24–32, 2007. doi:10.1109/CCC.2007.18.

8 Harry Buhrman and Ronald de Wolf. Communication complexity lower bounds by polynomials. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity (CCC)*, pages 120–130, 2001. doi:10.1109/CCC.2001.933879.

9 Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and markov-bernstein inequalities. *Inf. Comput.*, 243:2–25, 2015. Earlier version in ICALP'13. doi:10.1016/j.ic.2014.12.003.

10 Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting using low-discrepancy gadgets. *SIAM J. Comput.*, 50(1):171–210, 2021. doi:10.1137/19M1310153.

11 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudo-random properties. *Comput. Complex.*, 28(4):617–659, 2019. doi:10.1007/s00037-019-00190-7.

12 Hong-Bin Chen and Frank K. Hwang. A survey on nonadaptive group testing algorithms through the angle of decoding. *J. Comb. Optim.*, 15(1):49–59, 2008. doi:10.1007/s10878-007-9083-3.

13 Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 295–304. IEEE Computer Society, 2016.

14 Arkadii Georgievich Dyachkov and Vladimir Vasil'evich Rykov. A survey of superimposed code theory. *Problems of Control and Information Theory*, 12(4):1–13, 1983.

15 Peter Frankl and Norihide Tokushige. The Erdos-Ko-Rado theorem for integer sequences. *Comb.*, 19(1):55–63, 1999. doi:10.1007/s004930050045.

16 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory Comput.*, 16:1–30, 2020.

17 Dmitry Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity via max-conflict complexity. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 64:1–64:13, 2019.

18 Mika Göös and T. S. Jayram. A composition theorem for conical juntas. In *31st Conference on Computational Complexity (CCC)*, pages 5:1–5:16, 2016. doi:10.4230/LIPIcs.CCC.2016.5.

19 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM J. Comput.*, 45(5):1835–1869, 2016.

20 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM J. Comput.*, 47(6):2435–2450, 2018. doi:10.1137/16M1059369.

21 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. *SIAM J. Comput.*, 49(4), 2020. Earlier version in FOCS'17. doi:10.1137/17M115339X.

22 Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Structure of protocols for XOR functions. *SIAM J. Comput.*, 47(1):208–217, 2018. Earlier version in FOCS'16. doi:10.1137/17M1136869.

23 Kaave Hosseini, Shachar Lovett, and Grigory Yaroslavtsev. Optimality of linear sketching under modular updates. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

24 Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 526–535, 2007.

25 Sampath Kannan, Elchanan Mossel, Swagato Sanyal, and Grigory Yaroslavtsev. Linear sketching over f_2. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 8:1–8:37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

26 Hartmut Klauck. On quantum and probabilistic communication: Las vegas and one-way protocols. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC)*, pages 644–651, 2000. doi:10.1145/335305.335396.

27 Alexander Knop, Shachar Lovett, Sam McGuire, and Weiqiang Yuan. Log-rank and lifting for and-functions. In *Proceedings of 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 197–208. ACM, 2021. doi:10.1145/3406325.3450999.

28 Srijita Kundu. One-way quantum communication complexity with inner product gadget. *Electron. Colloquium Comput. Complex.*, 24:152, 2017. Comment #1 to TR17-152.

29 Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

30 Bruno Loff and Sagnik Mukhopadhyay. Lifting theorems for equality. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 126 of *LIPIcs*, pages 50:1–50:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

31 László Lovász and Michael E. Saks. Lattices, möbius functions and communication complexity. In *29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 81–90, 1988. doi:10.1109/SFCS.1988.21924.

32 Shachar Lovett. Communication is bounded by root of rank. *J. ACM*, 63(1):1:1–1:9, 2016. doi:10.1145/2724704.

33 Nikhil S. Mande, Suhail Sherif, and Swagato Sanyal. One-way communication complexity and non-adaptive decision trees. *CoRR*, abs/2105.01963, 2021. arXiv:2105.01963.

34 Ashley Montanaro. Nonadaptive quantum query complexity. *Inf. Process. Lett.*, 110(24):1110–1113, 2010. doi:10.1016/j.ipl.2010.09.009.

35 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago J. Theor. Comput. Sci.*, 2014, 2014.

36 Ashley Montanaro and Tobias Osborne. On the communication complexity of XOR functions. *CoRR*, abs/0909.3392, 2009. arXiv:0909.3392.

37 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000. URL: https://books.google.com/books?id=-s4DEy7o-a0C.

38 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Comb.*, 19(3):403–435, 1999. doi:10.1007/s004930050062.

39 Ben Reichardt. Reflections for quantum query algorithms. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 560–569, 2011.

40 Swagato Sanyal. One-way communication and non-adaptive decision tree. *Electron. Colloquium Comput. Complex.*, 24:152, 2017. URL: `https://eccc.weizmann.ac.il/report/2017/152`.

41 Swagato Sanyal. Fourier sparsity and dimension. *Theory of Computing*, 15(11):1–13, 2019. `doi:10.4086/toc.2019.v015a011`.

42 Alexander A. Sherstov. Making polynomials robust to noise. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*, pages 747–758, 2012.

43 Alexander A. Sherstov. Approximating the AND-OR tree. *Theory Comput.*, 9:653–663, 2013. `doi:10.4086/toc.2013.v009a020`.

44 Avishay Tal. Properties and applications of boolean function composition. In *Innovations in Theoretical Computer Science (ITCS)*, pages 441–454, 2013.

45 VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

46 Ronald de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002. `doi:10.1016/S0304-3975(02)00377-8`.

47 Hsin-Lung Wu. On the communication complexity of AND functions. *IEEE Transactions on Information Theory*, 2021.

48 Xiaodi Wu, Penghui Yao, and Henry S. Yuen. Raz-mckenzie simulation with the inner product gadget. *Electron. Colloquium Comput. Complex.*, 24:10, 2017. URL: `https://eccc.weizmann.ac.il/report/2017/010`.

49 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979. `doi:10.1145/800135.804414`.

50 Julius Žilinskas, Algirdas Lančinskas, and Mario R Guarracino. Pooled testing with replication as a mass testing strategy for the covid-19 pandemics. *Scientific Reports*, 11(1):1–7, 2021.

A Preliminaries

▶ **Definition A.1.** *For an integer $n \geq 2$ that is a power of 2, define the Addressing function, denoted* $\mathsf{ADDR}_n : \{0,1\}^{\log n + n} \to \{0,1\}$*, by*

$$\mathsf{ADDR}_n(x, y) = y_{\mathsf{bin}(x)},$$

where $\mathsf{bin}(x)$ *denotes the integer in* $[n]$ *whose binary representation is* x*. We refer to the* x*-variables as* addressing variables *and the* y*-variables as* target variables*.*

▶ **Definition A.2** (Non-adaptive parity decision tree complexity)**.** *Define the* non-adaptive parity decision tree complexity *of* $f : \{0,1\}^n \to \{0,1\}$*, denoted by* $\mathsf{NAPDT}(f)$*, to be the minimum number of parities such that* f *can be expressed as a function of these parities. In other words, the non-adaptive parity decision tree complexity of* f *equals the minimal number* k *for which there exists* $\mathcal{S} = \{\{S_1, \ldots, S_k\} : S_i \subseteq [n]$ *for all* $i \in [k]\}$ *such that the function value* $f(x)$ *is determined by the values* $\{\oplus_{j \in S_i} x_j : i \in [k]\}$ *for all* $x \in \{0,1\}^n$*.*

▶ **Definition A.3** (Non-adaptive AND decision tree complexity)**.** *Define the* non-adaptive AND decision tree complexity *of* $f : \{0,1\}^n \to \{0,1\}$*, denoted by* $\mathsf{NAADT}(f)$*, to be the minimum number of monomials such that* f *can be expressed as a function of these monomials. In other words, the non-adaptive AND decision tree complexity of* f *equals the minimal number* k *for which there exists* $\mathcal{S} = \{\{S_1, \ldots, S_k\} : S_i \subseteq [n]$ *for all* $i \in [k]\}$ *such that the function value* $f(x)$ *is determined by the values* $\{\mathsf{AND}_{S_i}(x) : i \in [k]\}$ *for all* $x \in \{0,1\}^n$*. We refer to such a set* $\mathcal{S}$ *as an* NAADT basis *for* f*.*

▶ **Definition A.4** (Randomized non-adaptive AND decision tree complexity). *A randomized non-adaptive AND decision tree T computing f is a distribution over non-adaptive AND decision trees with the property that* $\Pr[T(x) = f(x) \geq 2/3]$ *for all* $x \in \{0,1\}^n$*. The cost of T is the maximum cost of a non-adaptive AND decision tree in its support. Define the* randomized non-adaptive AND decision tree complexity *of* $f : \{0,1\}^n \to \{0,1\}$*, denoted by* $\mathsf{RNAADT}(f)$*, to be the minimum cost of a randomized non-adaptive AND decision tree that computes f.*

We refer the reader to [37] for the basics of quantum computing.

▶ **Definition A.5** (Quantum non-adaptive AND decision tree complexity). *A quantum non-adaptive AND decision tree of cost c is a query algorithm that works with a state space* $|S_1, \ldots, S_c\rangle|b\rangle|w\rangle$*, where each* $S_j \subseteq [n]$*,* $b \in \{0,1\}^c$ *and the last register captures a workspace of an arbitrary dimension. It is specified by a starting state $|\psi\rangle$ and a projective measurement* $\{\Pi, I - \Pi\}$*. For an input* $x \in \{0,1\}^n$*, the action of the non-adaptive query oracle* $O_x^{\otimes c}$ *is captured by its action on the basis states, described below.*

$$O_x^{\otimes c}|S_1, \ldots, S_c\rangle|b_1, \ldots, b_c\rangle|w\rangle \mapsto |S_1, \ldots, S_c\rangle|b_1 \oplus \mathsf{AND}_{S_1}(x), \ldots, b_c \oplus \mathsf{AND}_{S_c}(x)\rangle|w\rangle.$$

We use O_x to refer to this oracle since c is already unambiguously determined by the state space. The algorithm accepts x with probability $\|\Pi O_x|\psi\rangle\|^2$*.*

Define the quantum non-adaptive AND decision tree complexity *of* $f : \{0,1\}^n \to \{0,1\}$*, denoted by* $\mathsf{QNAADT}(f)$*, to be the minimum cost of a quantum non-adaptive AND decision tree that outputs the correct value of $f(x)$ with probability at least $2/3$ for all* $x \in \{0,1\}^n$*.*

The quantum non-adaptive query complexity, denoted $\mathsf{Q}_{\mathrm{dt}}^{\rightarrow}$, is defined similarly, the only difference being that the sets $S_1, \ldots, S_c$ are restricted to be singletons. Montanaro [34] observed that $\mathsf{Q}_{\mathrm{dt}}^{\rightarrow}(f) = \Omega(n)$ for all total Boolean functions $f : \{0,1\}^n \to \{0,1\}$ that depend on all input bits. Our proof of Theorem 4.9 uses ideas from their proof.

B Proof of Claim 4.4

In this section we prove Claim 4.4, restated below.

▷ Claim B.1 (Restatement of Claim 4.4). Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\mathsf{Pat}^{\mathsf{M}}(f) \leq 2^{(1-\Omega(1))\mathrm{spar}(f)}$.

Our proof of Claim 4.4 relies on the following observation about the structure of the Möbius support of any Boolean function.

▷ Claim B.2. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with Möbius support $\mathcal{S}_f$. For any two distinct sets $S, T \in \mathcal{S}_f$ there exists a set of "partners" $p(\{S,T\}) \subseteq \mathcal{S}_f$ such that

- $p(\{S,T\}) \neq \{S,T\}$,
- $|p(\{S,T\})| = 2$ if $S \cup T \notin \mathcal{S}_f$ and $|p(\{S,T\})| = 1$ if $S \cup T \in \mathcal{S}_f$, and
- $\bigcup_{U \in p(\{S,T\})} U = S \cup T$.

Proof. Let $\sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S$ be the Möbius expansion of f. Since f has range $\{0,1\}$, we know that $f = f^2$. However,

$$f^2 = \left(\sum_{S \in \mathcal{S}_f} \widetilde{f}(S)\mathsf{AND}_S\right)\left(\sum_{T \in \mathcal{S}_f} \widetilde{f}(T)\mathsf{AND}_T\right) = \sum_{W \subseteq [n]} \left(\sum_{S,T \subseteq [n] : S \cup T = W} \widetilde{f}(S)\widetilde{f}(T)\right)\mathsf{AND}_W.$$

Since the Möbius expansion of f is unique, we can compare the two expansions to see that for all sets $W \subseteq [n]$,

$$\widetilde{f}(W) = \sum_{S,T \subseteq [n] : S \cup T = W} \widetilde{f}(S)\widetilde{f}(T). \tag{12}$$

As a consequence we have the following structure. Let $S \neq T \in \mathcal{S}_f$ such that $S \cup T \notin \mathcal{S}_f$. Since $\widetilde{f}(S \cup T) = 0$, the summation corresponding to $W = S \cup T$ in Equation (12) must have at least one non-zero summand apart from $\widetilde{f}(S)\widetilde{f}(T)$. Hence there must exist $U \neq V \in \mathcal{S}_f$ such that $\{S, T\} \neq \{U, V\}$ and $U \cup V = S \cup T$. We choose an arbitrary such pair $\{U, V\}$ and define $p(\{S, T\}) = \{U, V\}$. For $S, T \in \mathcal{S}_f$ such that $S \cup T \in \mathcal{S}_f$, let $p(\{S, T\})$ be defined as $\{S \cup T\}$. It clearly satisfies the necessary conditions. ◁

▶ **Observation B.3.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with Möbius support $\mathcal{S}_f$. For any two distinct sets $S, T \in \mathcal{S}_f$, let $p(\{S, T\}) \subseteq \mathcal{S}_f$ be as in Claim B.2. Then for any pattern $P \in \{0,1\}^{\mathcal{S}_f}$,*

$$P_S \cdot P_T = \prod_{W \in p(\{S,T\})} P_W.$$

Proof. Let P be a pattern in $\{0,1\}^{\mathcal{S}_f}$. There must exist an $x \in \{0,1\}^n$ such that for all sets $W \in \mathcal{S}_f$, $P_W = \mathsf{AND}_W(x)$. Since $S \cup T = \bigcup_{W \in p(\{S,T\})} W$, we have $P_S \cdot P_T = \mathsf{AND}_{S \cup T}(x) = \prod_{W \in p(\{S,T\})} P_W$. ◀

Proof of Claim 4.4. We analyze the pattern complexity of f in iterations. To define these iterations, we define a sequence of subsets of $\mathcal{S}_f$, described in Algorithm 1.

Algorithm 1 Defining the Iterations.

```
Initialize $\mathcal{T}_0 \leftarrow \emptyset, i \leftarrow 0$.
while $|\mathcal{T}_i| \leq \mathrm{spar}(f) - 2$ do
    Choose $S, T$ with $S \neq T$ from $\mathcal{S}_f \setminus \mathcal{T}_i$.
    Set $\mathcal{T}_{i+1} \leftarrow \mathcal{T}_i \cup \{S, T\} \cup p(\{S, T\})$.
    Set $i \leftarrow i + 1$.
end
Set num_iterations $\leftarrow i$.
Set $\mathcal{T}_{\mathsf{num\_iterations}+1} \leftarrow \mathcal{S}_f$.
```

For $i \in \{0, \ldots, \mathsf{num\_iterations} + 1\}$, define the partial patterns

$$\mathcal{P}_i := \left\{ P \in \{0,1\}^{\mathcal{T}_i} : P = (\mathsf{AND}_S(x))_{S \in \mathcal{T}_i} \text{ for some } x \in \{0,1\}^n \right\}.$$

We now show that

$$\forall j \in \{0, \ldots, \mathsf{num\_iterations}\}, \; |\mathcal{P}_j| \leq \left(\frac{15}{16}\right)^j 2^{|\mathcal{T}_j|}. \tag{13}$$

We prove this by induction. Equation (13) is true when $j = 0$ since both sides are 1. Now let $i > 0$ and assume as our induction hypothesis that Equation (13) is true when $j = i - 1$. As our inductive step, we will prove that for every partial pattern $P \in \mathcal{P}_{i-1}$, the number of partial patterns $Q \in \mathcal{P}_i$ that extend P (in the sense that Q restricted to indices in $\mathcal{T}_{i-1}$ is equal to P) is at most $(15/16)2^{|\mathcal{T}_i| - |\mathcal{T}_{i-1}|}$. Since every partial pattern in $\mathcal{P}_i$ is an extension of a partial pattern in $\mathcal{P}_{i-1}$, this would imply that $|\mathcal{P}_i| \leq (15/16)2^{|\mathcal{T}_i| - |\mathcal{T}_{i-1}|}|\mathcal{P}_{i-1}|$. Along with our induction hypothesis, this will prove Equation (13) for $j = i$, and hence for all j.

To prove the inductive step, consider any partial pattern $P \in \mathcal{P}_{i-1}$. Let S, T be the sets chosen when constructing $\mathcal{T}_i$ from $\mathcal{T}_{i-1}$. We know from Observation B.3 that any partial pattern $Q \in \mathcal{P}_i$ must satisfy $Q_S \cdot Q_T = \prod_{W \in p(\{S,T\})} Q_W$. Consider the extension Q' of P that sets $Q'_W = 1$ for all $W \in p(\{S,T\})$ and $Q'_W = 0$ for all $W \in \{S,T\} \setminus p(\{S,T\})$. Clearly such a Q' does not satisfy $Q'_S \cdot Q'_T = \prod_{W \in p(\{S,T\})} Q'_W$. Hence of the $2^{|\mathcal{T}_i| - |\mathcal{T}_{i-1}|}$ possible extensions of P, at most $2^{|\mathcal{T}_i| - |\mathcal{T}_{i-1}|} - 1$ will be in $\mathcal{P}_i$. Since $|\mathcal{T}_i| - |\mathcal{T}_{i-1}| \leq 4$, we can conclude that

$$|\mathcal{P}_i| \leq (2^{|\mathcal{T}_i| - |\mathcal{T}_{i-1}|} - 1)|\mathcal{P}_{i-1}| \leq (15/16) 2^{|\mathcal{T}_i| - |\mathcal{T}_{i-1}|} |\mathcal{P}_{i-1}|.$$

This proves Equation (13).

Finally, note that the while loop in Algorithm 1 quits when $|\mathcal{T}_i| \geq \mathsf{spar}(f) - 1$. Hence $\mathsf{num\_iterations} \geq (\mathsf{spar}(f) - 1)/4$. If it quits with $|\mathcal{T}_i| = \mathsf{spar}(f)$, then Equation (13) implies that $\mathsf{Pat}^{\mathsf{M}}(f) \leq (15/16)^{(\mathsf{spar}(f)-1)/4} 2^{\mathsf{spar}(f)} \leq 1.02 \cdot 2^{0.98 \mathsf{spar}(f)}$. If it quits with $|\mathcal{T}_i| = \mathsf{spar}(f) - 1$, then each of the partial patterns in $\mathcal{P}_i$ can have at most two extensions to actual patterns of f. Hence even in this case $\mathsf{Pat}^{\mathsf{M}}(f) \leq 2.04 \cdot 2^{0.98 \mathsf{spar}(f)}$. ◁

In fact with a more careful analysis (see [33, Proof of Claim 4.4]) we obtain an upper bound of $\mathsf{Pat}^{\mathsf{M}}(f) \leq 2^{((\log 6)/3)\mathsf{spar}(f)+1} \approx 2^{0.86 \mathsf{spar}(f)+1}$.

C On Non-Adaptive AND Decision Trees for Symmetric Functions

Recall Theorem 4.14, restated below.

▶ **Theorem C.1** (Restatement of Theorem 4.14). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with* $\mathsf{switch}(f) = k < n/2$. *Then*

$$\mathsf{NAADT}(f) = O\left(\log^2 \binom{n}{k}\right).$$

The proof is via the probabilistic method. We construct a random family of $O\left(\log^2 \binom{n}{k}\right)$ many ANDs and argue that with non-zero probability, their evaluations on any input determine the function's value.

We require the following intermediate claim.

▷ Claim C.2. Let n be a positive integer, and let $1 \leq k < n/2$ be an integer. Then, there exists a collection $\mathcal{X}$ of $O\left(\log^2 \binom{n}{k}\right)$ many subsets of $[n]$ satisfying the following.

$$\forall i_1, \ldots, i_{k+1} \in [n], j \in [k+1], \exists X \in \mathcal{X} \text{ such that } i_j \in X, i_\ell \notin X \text{ for all } \ell \neq j. \tag{14}$$

Proof. Consider a random set $X \subseteq [n]$ chosen as follows: For each index $i \in [n]$ independently, include i in X with probability $1/(2k)$. Pick w many sets (where w is a parameter that we fix later) independently using the above sampling process, giving the multiset of sets $\mathcal{X} = \{X_1, \ldots, X_w\}$.

For fixed $i_1, \ldots, i_{k+1} \in [n]$, $j \in [k+1]$ and $t \in [w]$,

$$\Pr_{X_t}[i_j \in X_t \text{ and } i_\ell \notin X_t \text{ for all } \ell \neq j] = \frac{1}{2k} \cdot \left(1 - \frac{1}{2k}\right)^k \geq \frac{1}{2k \cdot e}, \tag{15}$$

where the last inequality uses the fact that $k \geq 1$ and the standard inequality that $1 - x \geq e^{-2x}$ for all $0 \leq x \leq 1/2$. Thus Equation (15) implies that for fixed $i_1, \ldots, i_{k+1} \in [n]$ and $j \in [k+1]$,

$$\Pr_{\mathcal{X}}[\nexists X \in \mathcal{X} : i_j \in X \text{ and } i_\ell \notin X \text{ for all } \ell \neq j] \leq \left(1 - \frac{1}{2k \cdot e}\right)^w \leq \exp(-w/(2ke)). \tag{16}$$

By a union bound over these "bad events" for all $i_1, \dots, i_{k+1} \in [n]$ and $j \in [k+1]$, we conclude that

$$\Pr_{\mathcal{X}}[\forall i_1, \dots, i_{k+1} \in [n] \text{ and } j \in [k+1],\ \exists X \in \mathcal{X} : i_j \in X \text{ and } i_\ell \notin X \text{ for all } \ell \neq j]$$
$$\geq 1 - \binom{n}{k+1} \cdot (k+1) \cdot \exp(-w/(2ke)). \tag{17}$$

We want to choose w such that this probability is greater than 0. Thus we require

$$1 > \binom{n}{k+1} \cdot (k+1) \cdot \exp(-w/(2ke))$$
$$\iff \exp(w/(2ke)) > (k+1) \cdot \binom{n}{k+1}$$
$$\iff w > 2ke \left(\log(k+1) + \log \binom{n}{k+1} \right).$$

Since $\binom{n}{j+1} \geq n > j+1$ for all $j \in \{1, 2, 3, \dots, n/2\}$ and $n > 2$, and since $\log \binom{n}{j} \geq j \log(n/j) \geq j$ for all $j \in \{1, 2, \dots, n/2\}$, it suffices to choose

$$w \geq 2e \log \binom{n}{k} \left(2 \log \binom{n}{k+1} \right). \tag{18}$$

By standard binomial inequalities we have $\log \binom{n}{k+1} \leq (k+1) \log(ne/(k+1))$, and $\log \binom{n}{k} > k \log(n/k)$. Next, since $k+1 \leq 2k$ for $k \geq 1$ and $ne/(k+1) < n^3/k^3$ for $k \in \{1, 2, \dots, n/2\}$, Equation (18) implies that it suffices to choose

$$w \geq 2e \log \binom{n}{k} \left(12 \log \binom{n}{k} \right).$$

For this choice of w, the RHS of Equation (17) is strictly positive. This proves the claim. ◁

Proof of Theorem 4.14. Let f be a symmetric function with $\mathsf{switch}(f) = k < n/2$, and let $\mathcal{X}$ be as in Claim C.2 with $|\mathcal{X}| = O\left(\log^2 \binom{n}{k}\right)$. We now show how $\mathcal{X}$ yields a NAADT for f. Without loss of generality assume that $f(x) = 0$ for all $|x| < n-k$ (if not, output 1 in place of 0 in the **Output** step of Algorithm 2 below).

Algorithm 2 NAADT for f.

```
Input: $x \in \{0,1\}^n$
  1. Let $\mathcal{X}$ be as obtained from Claim C.2.
  2. Query $\{\mathsf{AND}_X(x) : X \in \mathcal{X}\}$ to obtain a string $P_x \in \{0,1\}^{|\mathcal{X}|}$.
Output: $f(y)$ if $P_x = P_y$ for some $y$ with $|y| \geq n-k$, and 0 otherwise.
```

We show below that the following holds: $P_x \neq P_y$ for all $x \neq y \in \{0,1\}^n$ such that $|y| \geq n-k$. This would show correctness of the algorithm as follows:

- If $P_x = P_y$ for some $|y| \geq n-k$, then x must equal y by the above. In this case we output the correct value since we have learned x.
- If $P_x \neq P_y$ for any $|y| \geq n-k$, then $|x| < n-k$. Since f evaluates to 0 on all such inputs, we output the correct value in this case.

Let $x \neq y \in \{0,1\}^n$ be two strings such that $|y| \geq n-k$. Without loss of generality assume $|y| \geq |x|$ (else swap the roles of x and y above). Let $I_x, I_y \subseteq [n]$ denote the sets of indices where x and y take value 0, respectively. By assumption, $x \neq y$ and $|I_x| \geq |I_y|$. Thus there exists an index $i_x \in I_x \setminus I_y$.

Since $|I_y| \leq k$, by Claim C.2 there exists $X \in \mathcal{X}$ such that $i_x \in X$ and $X \cap I_y = \emptyset$. Thus, for this X we have

$$\mathsf{AND}_X(x) = 0, \qquad \mathsf{AND}_X(y) = 1.$$

Hence $P_x \neq P_y$, which proves the correctness of the algorithm and yields the theorem. ◀

▶ Remark C.3. The proof above in fact yields a NAADT of cost $O\left(\log^2 \binom{n}{k}\right)$ for any function $f : \{0,1\}^n \to \{0,1\}$ for which f is a constant on inputs of Hamming weight less than $n-k$ for some $k < n/2$ (in particular, f need not be symmetric on inputs of larger Hamming weight).

Isolation Schemes for Problems on Decomposable Graphs

Jesper Nederlof
Utrecht University, The Netherlands

Michał Pilipczuk
University of Warsaw, Poland

Céline M. F. Swennenhuis
Eindhoven University of Technology, The Netherlands

Karol Węgrzycki
Saarland University, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

The Isolation Lemma of Mulmuley, Vazirani and Vazirani [Combinatorica'87] provides a self-reduction scheme that allows one to assume that a given instance of a problem has a unique solution, provided a solution exists at all. Since its introduction, much effort has been dedicated towards derandomization of the Isolation Lemma for specific classes of problems. So far, the focus was mainly on problems solvable in polynomial time.

In this paper, we study a setting that is more typical for NP-complete problems, and obtain partial derandomizations in the form of significantly decreasing the number of required random bits. In particular, motivated by the advances in parameterized algorithms, we focus on problems on decomposable graphs. For example, for the problem of detecting a Hamiltonian cycle, we build upon the rank-based approach from [Bodlaender et al., Inf. Comput.'15] and design isolation schemes that use

- $\mathcal{O}(t\log n + \log^2 n)$ random bits on graphs of treewidth at most t;
- $\mathcal{O}(\sqrt{n})$ random bits on planar or H-minor free graphs; and
- $\mathcal{O}(n)$-random bits on general graphs.

In all these schemes, the weights are bounded exponentially in the number of random bits used. As a corollary, for every fixed H we obtain an algorithm for detecting a Hamiltonian cycle in an H-minor-free graph that runs in deterministic time $2^{\mathcal{O}(\sqrt{n})}$ and uses polynomial space; this is the first algorithm to achieve such complexity guarantees. For problems of more local nature, such as finding an independent set of maximum size, we obtain isolation schemes on graphs of treedepth at most d that use $\mathcal{O}(d)$ random bits and assign polynomially-bounded weights.

We also complement our findings with several unconditional and conditional lower bounds, which show that many of the results cannot be significantly improved.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Isolation Lemma, Derandomization, Hamiltonian Cycle, Exact Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.50

Related Version *Full Version*: https://arxiv.org/abs/2105.01465

Funding *Jesper Nederlof*: Supported by the project CRACKNP (grant agreement No 853234) that has received funding from the European Research Council (ERC).
Michał Pilipczuk: Supported by the project TOTAL (grant agreement No 677651) that has received funding from the European Research Council (ERC).
Céline M. F. Swennenhuis: Supported by the Netherlands Organization for Scientific Research under project no. 613.009.031b.
Karol Węgrzycki: Supported by the project TIPEA (grant agreement No 850979) that has received funding from the European Research Council (ERC).

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 50; pp. 50:1–50:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Isolation is a procedure that allows to single out a unique solution to a given problem within a possibly larger solution space, thus effectively reducing the original problem to a variant where one may assume that if a solution exists, then there is a unique one. The classic Isolation Lemma of Mulmuley, Vazirani and Vazirani [26] can be used to achieve this at the cost of allowing randomization. In complexity theory, isolation is used to show that hard problems are not easier to solve on instances with unique solutions [35]. This idea has found numerous applications ranging from structural results in complexity theory (e.g. $\mathsf{NL/poly} \subseteq \oplus\mathsf{L/poly}$ [37] or $\mathsf{NL/poly} = \mathsf{UL/poly}$ [32]) to the design of parallel algorithms [26, 22, 17, 34].

Since obtaining a general derandomization of the Isolation Lemma is impossible by counting arguments [4, 8, 1], it is natural to ask whether the isolation step can be derandomized for specific problems with explicit representation. In this context, there has recently been an exciting progress in isolation for perfect matchings [2, 7, 13, 21, 3, 22], which culminated in an isolation scheme that uses $\mathcal{O}(\log^3 n)$ random bits, implying a quasi-NC algorithm for detecting a perfect matching [34].

In contrast to this, derandomization of isolation procedures for NP-complete problems is relatively less studied, and not because of a lack of motivation: Many contemporary fixed-parameter algorithms rely on the Isolation Lemma [25, 28, 5, 23, 24, 11, 38]. Usually, the isolation procedure is the only subroutine requiring randomness. Many of the algorithms mentioned above apply the Isolation Lemma in combination with a decomposition-based method such as Divide&Conquer or dynamic programming. This motivates us to study the following:

▶ **Main Question.** *How much randomness is required for isolating problems with decomposable structure?*

More concretely, we focus on graph problems where the underlying graph is *decomposable*, in the sense that it can be decomposed using small separators. Examples of such graphs are planar graphs or graphs of bounded treewidth. It is well-known that for many NP-complete problems, the nice structure of such graphs can be leveraged to solve these problems faster than in general graphs. We show that a similar phenomenon occurs when one considers the amount of randomness needed to isolate a single solution.

The model for isolation schemes. Suppose U is a finite set and $\omega\colon U \to \mathbb{N}$ is a weight function. For $X \subseteq U$ we write $\omega(X) := \sum_{e\in X} \omega(e)$. For a set family $\mathcal{F} \subseteq 2^U$ we say that ω *isolates* $\mathcal{F}$ if there is exactly one set $S \in \mathcal{F}$ such that $\omega(S)$ is the minimum possible among the weights of the sets in $\mathcal{F}$. The classic Isolation Lemma of Mulmuley et al. [26] states that a weight function $\omega\colon U \to \{1, \ldots, 2|U|\}$ chosen uniformly at random isolates any family $\mathcal{F} \subseteq 2^U$ with probability at least $\frac{1}{2}$. Note that sampling such ω requires $\mathcal{O}(|U| \log |U|)$ random bits.

Most of our isolation schemes work in a very restricted model inspired by the discussion above, which we explain now. Intuitively, the scheme is not aware of the graph or its decomposition, but is only aware of the vertex count of the graph and the relevant width parameter, such as the treewidth or treedepth.

Formally, a *vertex selection problem* is a function $\mathcal{P}$ that maps every graph G to a family $\mathcal{P}(G) \subseteq 2^{V(G)}$ consisting of subsets of the vertex set of G. Edge selection problems are defined analogously: $\mathcal{P}(G)$ consists of subsets of $E(G)$. For example, we could define a vertex selection problem $\mathsf{MIS}(\cdot)$ that maps every graph G to the family $\mathsf{MIS}(G)$ comprising all maximum-size independent sets in G, or an edge selection problem $\mathsf{HC}(\cdot)$ that maps every graph G to the family $\mathsf{HC}(G)$ comprising all (edge sets of) Hamiltonian cycles in G. Further, let $\mathcal{C}$ be a class of graphs, that is, a set of graphs that is invariant under isomorphism. For instance, $\mathcal{C}$ could be the class of planar graphs, or the class of graphs of treewidth at most k, for any fixed k. Then our definition of an isolation scheme reads as follows (here, we write $[n] \coloneqq \{1, \ldots, n\}$):

▶ **Definition 1.** *For a graph class $\mathcal{C}$, we say that a vertex selection problem $\mathcal{P}$ admits an* isolation scheme *on $\mathcal{C}$ with $\log \ell$ random bits and maximum weight W if for every $n \in \mathbb{N}$ there exist weight functions $\omega_1, \ldots, \omega_\ell \colon [n] \to [W]$ such that for every $G \in \mathcal{C}$ with vertex set $[n]$, ω_i isolates $\mathcal{P}(G)$ for at least half of the indices $i \in [\ell]$.*

Isolation schemes for edge selection problems are defined analogously: the weight functions $\omega_1, \ldots, \omega_\ell$ have domain $[m]$ and should assign weights to all the edges in m-edge graphs in $\mathcal{C}$, where the edges are assumed to be enumerated with numbers in $[m]$.

The two main parameters of interest for isolation schemes will be the number of *random bits*, which is defined as $\log \ell$, and the *maximum weight*, defined as the maximum value that any of the functions ω_i may take. Although Definition 1 only assumes the *existence* of suitable weight functions, all the isolation schemes proposed in this paper are extremely simple and can be used as an effective derandomization tool.

1.1 Our contribution

In the following discussion we restrict attention to Hamiltonian cycles and maximum-size independent sets for concreteness, that is, to the edge- and vertex-selection problems $\mathsf{HC}(\cdot)$ and $\mathsf{MIS}(\cdot)$ described above. However, our techniques have a wider applicability, which we comment on throughout the presentation. On a very high level, the natural idea that permeates all our arguments is to reduce the randomness using Divide&Conquer along small separators: If a separator X splits the given graph G in a balanced way, then the same random bits can be reused in each part of $G - X$.

Isolation schemes for Hamiltonian cycles. We first consider the problem of detecting a Hamiltonian cycle, since it represents an important class of connectivity problems such as Steiner Tree or k-Path. For these problems, the Isolation Lemma has been particularly useful in the design of parameterized algorithms [25, 28, 5, 23, 24, 11, 38]. Our first results concerns general graphs.

▶ **Theorem 2.** *There is an isolation scheme for Hamiltonian cycles in undirected graphs that uses $\mathcal{O}(n)$ random bits and assigns weights upper bounded by $2^{\mathcal{O}(n)}$.*

Observe that in an n-vertex graph there can be as many as $n!$ different Hamiltonian cycles. Hence, the application of the general-usage isolation scheme of Chari et al. [8] would give an isolation scheme for Hamiltonian cycles in general graphs that uses $\mathcal{O}(\log(n!)) = \mathcal{O}(n \log n)$ random bits. Note that as proved in [8], isolating a family $\mathcal{F}$ over a universe of size n requires $\Omega(\log |\mathcal{F}| + \log n)$ random bits in general, hence the shaving of the $\log n$ factor reported in Theorem 2 required a problem-specific insight into the family of Hamiltonian cycles in a graph. This insight is provided by the *rank-based approach*, a technique introduced in the context of detecting Hamiltonian cycles in graphs of bounded treewidth [6]. The fact that this works is unexpected because all known methods for derandomizing Hamiltonian cycle require at least exponential space (see [6] for overview).

Let us note that isolation of Hamiltonian cycles was used by Björklund [5] in his $\mathcal{O}(1.657^n)$-time algorithm for detecting a Hamiltonian cycle in an undirected graph. This algorithm is randomized due to the usage of the Isolation Lemma, and derandomizing it, even within time complexity $\mathcal{O}((2-\varepsilon)^n)$ for any $\varepsilon > 0$, is a major open problem. While the constant hidden in the $\mathcal{O}(\cdot)$ notation used in Theorem 2 is too large to allow exploring the whole space of random bits within time $\mathcal{O}((2-\varepsilon)^n)$, in principle we show that the amount of randomness needed is of the same magnitude as would be required for derandomization of the algorithm of Björklund.

Next, we show that in the setting of graphs of bounded treewidth the amount of randomness can be reduced dramatically, to a polylogarithm in n.

▶ **Theorem 3.** *For every $t \in \mathbb{N}$, there is an isolation scheme for Hamiltonian cycles in graphs of treewidth at most t that uses $\mathcal{O}(t \log n + \log^2(n))$ random bits and assigns weights upper bounded by $2^{\mathcal{O}(t \log n + \log^2 n)}$.*

The proof of Theorem 3 fully exploits the idea of using small separators to save on randomness. It also uses the rank-based approach to shave off a $\log t$ factor in the number of random bits.

Finally, we use the separator properties of H-minor free graphs to prove the following.

▶ **Theorem 4.** *For every fixed H, there is an isolation scheme for Hamiltonian cycles in H-minor-free graphs that uses $\mathcal{O}(\sqrt{n})$ random bits and assigns weights upper bounded by $2^{\mathcal{O}(\sqrt{n})}$.*

Recently, [28] presented a randomized algorithm for detecting a Hamiltonian cycle in a graph of treedepth at most d that works in time $2^{\mathcal{O}(d)} \cdot (W+n)^{\mathcal{O}(1)}$ time and uses polynomial space; here, W is the maximum weight assigned by isolation scheme[1]. The only source of randomness in the algorithm of [28] is the Isolation Lemma. Since H-minor free graphs have treedepth $\mathcal{O}(\sqrt{n})$, we can use the isolation scheme of Theorem 4 to derandomize this algorithm, thus obtaining the following result.

▶ **Theorem 5.** *For every fixed H, there is a deterministic algorithm for detecting a Hamiltonian cycle in an H-minor-free graph that runs in time $2^{\mathcal{O}(\sqrt{n})}$ and uses polynomial space.*

To the best of our knowledge, this is the first application of a randomness-efficient isolation scheme for a full derandomization of an exponential-time algorithm without a significant loss on complexity guarantees. Further, we are not aware of any previous algorithms that would simultaneously achieve determinism, running time $2^{\mathcal{O}(\sqrt{n})}$, and polynomial space complexity, even in the setting of planar graphs[2]. Finally, let us note that the algorithm of Theorem 5 does not rely on any topological properties of H-minor-free graphs: the existence of balanced separators of size $\mathcal{O}(\sqrt{n})$ is the only property we use.

MSO-definable problems on graphs of bounded treewidth. We observe that the approach used in the proof of Theorem 3 relies only on finite-state properties of the Hamiltonian Cycle problem on graphs of bounded treewidth. The range of problems enjoying such properties is much wider and encompasses all problems definable in CMSO_2: the Monadic

[1] They did not consider the weighted case, but the statement is implied by a standard extension, see the full version of this paper [27] for details.

[2] Deterministic $2^{\mathcal{O}(\sqrt{n})}$-time algorithms were previously known, but all of these use exponential space [6, 18].

Second-Order logic with modular counting predicates. Consequently, we can lift the proof of Theorem 3 to a generic reasoning that yields an analogous result for every CMSO_2-definable problem. This proves the following (see the full full version of this paper [27] for definitions).

▶ **Theorem 6.** *Let $\mathcal{P}$ be a CMSO_2-definable edge (or vertex) selection problem. There exists a computable function f such that for every $k \in \mathbb{N}$, $\mathcal{P}$ admits an isolation scheme on graphs of treewidth at most k that uses $R := f(k) \cdot \log n + \mathcal{O}(\log^2 n)$ random bits and assigns weights upper bounded by 2^R.*

Lower bounds. We show that a significant improvement of the parameters in the isolation schemes presented above is unlikely. First, a counting argument shows that the $\log n$ factor is necessary.

▶ **Theorem 7.** *There does not exist an isolation scheme for Hamiltonian cycles on graphs of treewidth at most 4 that uses $o(\log n)$ random bits and polynomially bounded weights.*

Using similar constructions we also provide analogous $\Omega(\log n)$ lower bounds for isolating other families of combinatorial objects related to NP-hard problems, such as maximum independent sets, minimum Steiner trees, and minimum maximal matchings. These lower bounds hold even in graphs of bounded *treedepth*, which is a more restrictive setting than bounded treewidth.

We also show using existing reductions that a significant improvement over the scheme of Theorem 2 would imply a surprising partial derandomization of isolation schemes for SAT.

▶ **Theorem 8.** *Suppose there is an isolation scheme for Hamiltonian cycles in undirected graphs that uses $o(n)$ random bits and polynomially bounded weights. Then there is a randomized polynomial-time reduction from* SAT *to* Unique SAT *that uses $o(n)$ random bits, where n is the number of variables.*

Observe that since an n-vertex graph has treewidth at most $n-1$, Theorem 8 also implies that in Theorem 3 one cannot expect reducing the number of random bits to $o(t)$. However, we stress that the lower bounds of Theorems 7 and 8 are not completely tight with respect to the upper bounds of Theorems 2 and 3, because the latter allow superpolynomial weights. It remains open whether the weights used by the schemes of Theorems 2, 3, and 4 can be reduced to polynomial.

In the full version of this paper [27] we further discuss consequences of the hypothetical existence of a polynomial-time reduction from SAT to Unique SAT that would use $o(n)$ random bits.

Level-aware isolation schemes for independent sets. In the light of the $\Omega(\log n)$ lower bound of Theorem 7, we consider a relaxation of the model from Definition 1, where the graph is provided together with an *elimination forest* (a decomposition notion suited for the graph parameter *treedepth*), and the weight of a vertex may depend both on the vertex' identifier and its level in the elimination forest. We demonstrate that in this relaxed model, the $\Omega(\log n)$ lower bound can be circumvented.

▶ **Definition 9.** *We say that vertex selection problem $\mathcal{P}$ admits a* level-aware isolation scheme *if for all $n, d \in \mathbb{N}$ there exist functions $\omega_1, \ldots, \omega_\ell \colon [n] \times [d] \to \mathbb{N}$ such that for every graph G on vertex set $[n]$ and elimination forest F of G of height at most d, at least half of the functions $\omega_1, \ldots, \omega_\ell$ isolate $\mathcal{P}(G)$. Here, when evaluating ω_i on a vertex $u \in [n]$, we apply ω_i to u and the index of the level of u in F.*

▶ **Theorem 10.** *For every $d \in \mathbb{N}$, there is a level-aware isolation scheme for maximum-size independent sets in graphs of treedepth at most d that uses $\mathcal{O}(d)$ random bits and assigns weights bounded by $\mathcal{O}(n^6)$.*

In the proof of Theorem 10 we describe an abstract condition, dubbed the *exchange property*, which is sufficient for the argument to go through. This property is enjoyed also by other families of combinatorial objects defined through constraints of local nature, such as minimum dominating sets or minimum vertex covers. Therefore, we can prove analogous isolation results for those families as well.

Also, in the full version of this paper [27] we discuss a similar reasoning for edge-selection problems on the example of maximum matchings, achieving a level-aware isolation scheme that uses $\mathcal{O}(d \log n)$ random bits and assigns weights bounded by $n^{\mathcal{O}(\log n)}$. This provides another natural class of graphs where isolation-based algorithms for finding a maximum matching can be derandomized (see [2, 7, 13, 21]).

We summarize our results with Table 1.

Table 1 Summary of our results based on Theorems 2-10.

Problem	Random Bits	Max Weight	Graph Class
Hamiltonian Cycle	$\mathcal{O}(n)$ $\Omega(n)$	$2^{\mathcal{O}(n)}$ poly(n)	General Graphs
	$\mathcal{O}(\sqrt{n})$ $\Omega(\sqrt{n})$	$2^{\mathcal{O}(\sqrt{n})}$ poly(n)	H-minor free graphs
	$\mathcal{O}(t\log(n) + \log^2(n))$ $\Omega(t + \log(n))$	$n^{\mathcal{O}(t+\log(n))}$ poly(n)	Treewidth t graphs
CMSO$_2$	$f(t)\log(n) + \mathcal{O}(\log^2(n))$	$n^{f(t)+\mathcal{O}(\log(n))}$	Treewidth t graphs
Max Independent Set	$\mathcal{O}(d)$ $\Omega(d)$	poly(n) poly(n)	Treedepth d graphs

1.2 Organization

In Section 2 we provide preliminaries. Section 3 is dedicated to the formal proof of Theorem 2. In Appendix A, we formally proof Theorem 3. We finish the main part of the paper with possible directions for further research in Section 4.

In the full version of this paper [27] we include the formal proofs of Theorem 4, Theorem 5 and the general CMSO$_2$-result of Theorem 6. The full version [27] also includes the lower bounds from Theorem 7 and Theorem 8, as well as the level-aware isolation schemes for local vertex (respectively, edge) selection problems.

2 Preliminaries

Notation. For an integer k, we write $[k] := \{1, \ldots, k\}$. We use standard graph notation: $V(G)$ and $E(G)$ respectively denote the vertex set and the edge set of a graph G, for $X \subseteq V(G)$ the *closed neighborhood* $N_G[X]$ is X plus all the neighbors of vertices of X, and the *open neighborhood* is $N_G(X) := N_G[X] \setminus X$.

Hashing modulo primes. The following standard hashing lemma that dates back to the work of Fredman, Komlós, and Szemerédi [19], will be the main source of randomness in our isolation schemes.

▶ **Lemma 11** (FKS hashing lemma [19]). *Let $S \subseteq \{0, 1, \ldots, 2^n\}$ be a set of k integers, where $n, k \geqslant 1$. Suppose that p is a prime number chosen uniformly at random among prime numbers in the range $\{1, \ldots, M\}$, where $M \geqslant 2$. Then*

$$\mathbb{P}\left[x \not\equiv y \bmod p \quad \textit{for all} \quad x, y \in S, x \neq y\right] \geqslant 1 - \frac{nk^2}{\sqrt{M}}.$$

Proof. Let

$$R := \prod_{x,y \in S, x \neq y} |x - y|.$$

Note that $R \leqslant 2^{n \cdot \binom{k}{2}}$. This implies that R may have at most $n \cdot \binom{k}{2}$ different prime divisors. On the other hand, from the prime number theorem it follows that $\pi(M) \in \Omega(\frac{M}{\log M})$, where $\pi(M)$ denotes the number of primes in the range $\{1, \ldots, M\}$. In fact, using a more precise estimate of Rosser [33], for $M \geqslant 17$ we have $\pi(M) \geqslant \frac{M}{\ln M}$. For $2 \leqslant M \leqslant 17$ a direct check shows that $\pi(M) \geqslant \sqrt{M}/2$. Since $\frac{M}{\ln M} \geqslant \sqrt{M}/2$ for all $M \geqslant 2$, we conclude that the probability that a random prime in the range $\{1, \ldots M\}$ is not among the at most $n \cdot \binom{k}{2}$ prime divisors of R is at least

$$1 - \frac{n \cdot \binom{k}{2}}{\sqrt{M}/2} \geqslant 1 - \frac{nk^2}{\sqrt{M}}. \qquad ◀$$

Graph decompositions. A *rooted forest* is directed acyclic graph F where every node x has at most one outneighbor, called the *parent* of x. A *root* is a node with no parent. If a node y is reachable from x by a directed path, then we write $y \preceq_F x$ and say that y is an *ancestor* of x and x is a *descendant* of y. Note that every vertex is considered its own ancestor and descendant. For $x \in V(F)$, we write

$$\mathsf{tail}_F[x] := \{y \colon y \preceq_F x\}, \qquad \mathsf{subtree}_F[x] := \{z \colon z \succeq_F x\},$$
$$\mathsf{tail}_F(x) := \mathsf{tail}_F[x] \setminus \{x\}, \qquad \mathsf{subtree}_F(x) := \mathsf{subtree}_F[x] \setminus \{x\}.$$

The *level* of a node x in F, denoted $\mathrm{lvl}_F(x)$, is the number of its strict ancestors, that is, $|\mathsf{tail}_F(x)|$. Note that roots have level 0. The *height* of a forest F is the maximum level among its nodes, plus 1. If the forest F is clear from the context, then we may omit it in the above notation.

An *elimination forest* of a graph G is a rooted forest F with $V(F) = V(G)$ such that for every edge uv of G, either u is an ancestor of v in F or vice versa. The *treedepth* of a graph G is the least possible height of an elimination forest of G. Treedepth as a graph parameter plays a central role in the structural theory of sparse graphs, see [29, Chapters 6 and 7]. It also has several applications in parameterized complexity and algorithm design [9, 15, 20, 28, 30, 31], as well as exhibits interesting combinatorial properties [9, 12, 14] and connections to descriptive complexity theory [16]. We refer to the introductory sections of the above works for a wider discussion.

A *tree decomposition* of a graph G is a pair $\mathbb{T} = (T, \beta)$, where T is an (unrooted) tree and $\beta \colon V(T) \to 2^{V(G)}$ is a function that assigns to each node $x \in V(T)$ its *bag* $\beta(x) \subseteq V(G)$ so that the following two conditions are satisfied:

- for each $u \in V(G)$, the set $\{x\colon u \in \beta(x)\}$ induces a nonempty and connected subtree of T; and
- for each $uv \in E(G)$, there exists $x \in V(T)$ such that $\{u, v\} \subseteq \beta(x)$.

The *width* of $\mathbb{T}$ is $\max_{x \in V(T)} |\beta(x)| - 1$ and the *treewidth* of G is the minimum possible width of a tree decomposition of G. It is easy to see that the treedepth of a graph is at least its treewidth plus one. Conversely, the treewidth is upper bounded by the treedepth times the logarithm of the vertex count [29].

For surgery on tree decompositions we will use the following definition and standard lemma.

▶ **Definition 12** (Segment of a tree). *For an unrooted tree T, a* segment *of T is a nonempty and connected subtree I of T such that there are at most two vertices of I that have a neighbor outside of I. The set of those at most two vertices is the* boundary *of I, and is denoted by ∂I. The* size *of I is equal to $|E(I)|$.*

▶ **Lemma 13.** *Let T be an unrooted tree and let I be a segment of T of size $\ell \geqslant 2$. Then there are at most 5 segments $I_1, \ldots, I_t$ of T ($t \leqslant 5$), each of size at most $\ell/2$, such that segments $I_1, \ldots, I_t$ have pairwise disjoint edge sets and $E(I_1) \cup \ldots \cup E(I_t) = E(I)$.*

Proof. For each edge $xy \in E(I)$, let $I_{y,x}$ and $I_{x,y}$ be the connected components of $I - xy$ that contain x and y, respectively. Let $\vec{I}$ be the orientation of I where each edge xy is oriented towards x if $|E(I_{y,x})| > |E(I_{x,y})|$ and towards y if $|E(I_{y,x})| < |E(I_{x,y})|$; in case $|E(I_{y,x})| = |E(I_{x,y})|$, the edge xy is oriented in any way. Since I has ℓ edges and $\ell + 1$ nodes, there is a node z of I that has outdegree 0 in $\vec{I}$. This means that for every neighbor x of z, we have $|E(I_{z,x})| \leqslant |E(I_{x,z})|$, implying $|E(I_{z,x})| < \ell/2$. Denote $I_x \coloneqq I_{z,x}$ and let $\widehat{I}_x$ be I_x with the edge xz added.

We first argue that I can be edge-partitioned into at most 3 subtrees (not necessarily segments), each with at most $\ell/2$ edges. Consider first the corner case when there exists a neighbor x of z such that $\widehat{I}_x$ has more than $\ell/2$ edges. Then both $I_x = I_{z,x}$ and $I_{x,z}$ have exactly $\frac{\ell-1}{2}$ edges each, so we can partition I into $I_{z,x}$, $I_{x,z}$, and a separate subtree consisting only of the edge xz. This case being resolved, we can assume that each tree $\widehat{I}_x$ has at most $\ell/2$ edges. Starting with the set of trees $\mathcal{T} \coloneqq \{\widehat{I}_x\colon x \text{ is a neighbor of } z\}$, iteratively apply the following procedure: take two trees from $\mathcal{T}$ with the smallest edge counts, and replace them with their union, provided this union has at most $\ell/2$ edges. The procedure stops when this assertion fails to be satisfied. Observe that the procedure can be carried out as long as $|\mathcal{T}| \geqslant 4$, for then the two trees from $\mathcal{T}$ that have the smallest edge counts together include at most half of the edges of I. Therefore, at the end we obtain the desired edge-partition of I into at most three subtrees.

All in all, in both cases we edge-partitioned I into at most three subtrees, each having at most $\ell/2$ edges. Since $|\partial I| \leqslant 2$, it is easy to see that all of those subtrees are already segments (i.e. have boundaries of size at most 2) apart from at most one, say J, which may have a boundary of size 3. Supposing that J exists, let $\partial J = \{a, b, c\}$. Then there exists a node d of J such that every connected component of $J - d$ contains at most one of the vertices a, b, c. It is now straightforward to edge-partition J into three trees so that the boundary of each of them consists of d and one of the vertices a, b, c. Thus, replacing J with those three segments yields an edge-partition of I into at most 5 segments, each with at most $\ell/2$ edges. ◀

3 Isolating Hamiltonian cycles

In this section we prove Theorem 2. We begin by defining *configurations* for Hamiltonian cycles, which reflect the states of a natural dynamic programming algorithm for detection of a Hamiltonian cycle in a bounded-treewidth graph. Then we use the rank-based approach to bound the number of *minimum weight compliant edge sets* (see Theorem 19). This technical result captures the essence of the rank-based approach and will be used in all subsections that follow. Next, we prove Theorem 2 in Section 3.3. In Appendix A we also include the full proof of Theorem 3.

3.1 Configurations for Hamiltonian cycles

Let us fix a graph G. An edge set $S \subseteq E(G)$ is called a *partial solution* if every vertex of G is incident to at most two edges of S and S has no cycles. The following notion of a *configuration* describes the behavior of a partial solution with respect to a set of vertices.

▶ **Definition 14** (Configurations)**.** *For $X \subseteq V(G)$, we define the set of* configurations $\mathsf{conf}(X)$ *on X as:*

$$\{\,(V_0, V_1, V_2, M)\;:\;(V_0, V_1, V_2) \textit{ is a partition of } X \textit{ and } M \textit{ is a perfect matching on } V_1\,\}.$$

Given a subgraph H of G, one can view the configurations on $X \subseteq V(H)$ as all possible different ways that a partial solution may behave on X. A vertex is then in the set V_i if it is incident to exactly i edges of the partial solution. The matching M on V_1 describes the endpoints of each path in the partial solution. This intuition is formalized in the following definition.

▶ **Definition 15.** *Let $X \subseteq V(G)$ be a set of vertices of G and let $S \subseteq E(G)$ be a partial solution. Then define the* configuration of S on X *as $c_X(S) := (V_0, V_1, V_2, M) \in \mathsf{conf}(X)$, where*

- $V_0 := \{v \in X : v$ *is not incident to any edge of* $S\}$,
- $V_1 := \{v \in X : v$ *is incident to exactly one edge of* $S\}$,
- $V_2 := \{v \in X : v$ *is incident to exactly two edges of* $S\}$,
- $M := \{\{u, v\} \in \binom{V_1}{2} :$ *there is a path with edges from* S *connecting* u *and* $v\}$.

We omit X in the notation and write $c(S)$ when X is clear from context.

Note that in the above definition M is indeed a matching, because each $v \in V_1$ is connected to exactly one $u \in V_1$ through S, as any partial solution covers each vertex at most twice. For an example of deriving $c_X(S)$ from a partial solution S, see Figure 1.

We can use configurations to tell whether two partial solutions together form a Hamiltonian cycle. Let H be a subgraph of G and let $X \subseteq V(H)$. Assume that there exists a partial solution S that visits only vertices from $(V(G)\setminus V(H))\cup X$, where every vertex of $V(G)\setminus V(H)$ is visited exactly twice. Then we only need to know $c_X(S)$ to determine which partial solutions $S' \subseteq E(H)$ would combine with S to a Hamiltonian cycle in G. We say that any such partial solution is *compliant* with $c_X(S)$, as expressed formally in the next definition.

▶ **Definition 16** (Compliant partial solution)**.** *For a graph H let $X \subseteq V(H)$. A configuration $c = (V_0, V_1, V_2, M) \in \mathsf{conf}(X)$ and a partial solution $S \subseteq E(H)$ are* compliant *if $S \cap M = \emptyset$ and $S \cup M$ forms a Hamiltonian cycle on $V(H) \setminus V_2$.*

See Figure 2 for an example of a compliant partial solution.

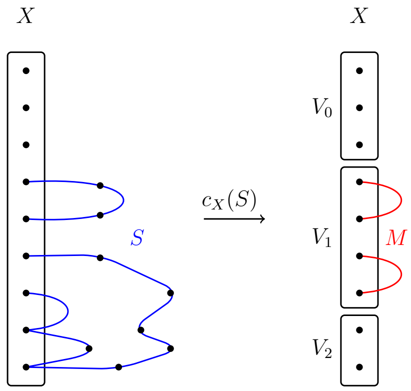


Figure 1 Example partial solution S and its configuration $c_X(S) = (V_0, V_1, V_2, M)$ on a set X.

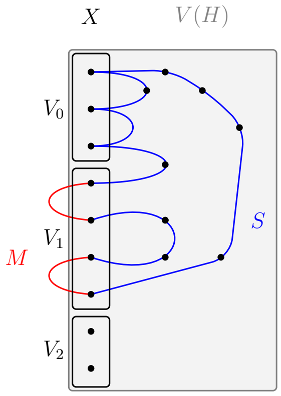


Figure 2 Example compliant partial solution S for a configuration $c = (V_0, V_1, V_2, M) \in \mathsf{conf}(X)$.

In the sequel we will be trying to argue that some weight function ω is isolating the family of Hamiltonian cycles in the given graph G with high probability. In all cases this will be done by induction on larger and larger subgraphs of G, where at each point we argue that a suitable family of partial solutions is isolated with high probability. The following definition facilitates this discussion.

▶ **Definition 17** (Minimum weight compliant partial solution). *Let H be a subgraph of G, $X \subseteq V(H)$, $c \in \mathsf{conf}(X)$, and let $\omega \colon E(G) \to \mathbb{N}$ be a weight function on the edges of G. Then we define the set $\mathsf{Min}(\omega, H, c)$ of* minimum weight partial solutions compliant with c *as the set of those partial solutions $S \subseteq E(H)$ that*

- *are compliant with c, and*
- *subject to the above, have the smallest possible weight $\omega(S)$.*

3.2 Rank-based approach

We will use the *rank-based approach*, introduced by Cygan et al. in [10], as a tool in our analysis of isolation schemes. Let X be a set of vertices. Then define the *compatibility matrix* $\mathcal{H}_X$ as the matrix with entries indexed by $\mathcal{H}_X[M_1, M_2]$ for M_1, M_2 perfect matchings on X, where

$$\mathcal{H}_X[M_1, M_2] = \begin{cases} 1 & \text{if } M_1 \cup M_2 \text{ is a simple cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathcal{H}_X[M_1, M_2]$ has $2^{\mathcal{O}(|X| \log |X|)}$ rows and columns. The crux of the rank-based approach is that in spite of that, this matrix has a small rank over the two-element field $\mathbb{F}_2$.

▶ **Theorem 18** (Rank-based approach,[10]). *For any set X, the rank of $\mathcal{H}_X$ over $\mathbb{F}_2$ is equal to $2^{|X|/2-1}$.*

We use Theorem 18 to prove that the total number of minimum weight compliant solutions is always relatively small, no matter what the weight function is. The following statement will be reused several times in the sequel. Note that a trivial cardinality argument would yield an upper bound of the form $2^{\mathcal{O}(|X| \log |X|)}$; the point of the rank-based approach is to reduce this to $2^{\mathcal{O}(|X|)}$.

▶ **Theorem 19.** *Let G be a graph, $X \subseteq V(G)$, and $\omega\colon V(G) \to \mathbb{N}$ be a weight function such that for all $c \in \mathsf{conf}(X)$, we have $|\mathsf{Min}(\omega, G, c)| \leqslant 1$. Then*

$$\left|\bigcup_{c\in\mathsf{conf}(X)} \mathsf{Min}(\omega, G, c)\right| \leqslant 2^{\mathcal{O}(|X|)}.$$

Proof. Let $K := \bigcup_{c\in\mathsf{conf}(X)} \mathsf{Min}(\omega, G, c)$ and let $C := \{c(S) \colon S \in K\}$.

We first verify that $|C| = |K|$. By construction, we have $|C| \leqslant |K|$. Assume for contradiction that $|C| < |K|$. Then there are two different partial solutions $S_1, S_2 \in K$ such that $c(S_1) = c(S_2)$. By construction and the assumptions, there are two different configurations $d_1, d_2 \in \mathsf{conf}(X)$ such that $\mathsf{Min}(\omega, G, d_1) = \{S_1\}$ and $\mathsf{Min}(\omega, G, d_2) = \{S_2\}$. However, since $c(S_1) = c(S_2)$, it follows that for any configuration $d \in \mathsf{conf}(X)$, S_1 is compliant with d if and only if S_2 is compliant with d. In particular, S_1 is compliant with d_2 and S_2 is compliant with d_1. This implies that $\omega(S_1) = \omega(S_2)$ and $S_2 \in \mathsf{Min}(\omega, G, d_1)$ and $S_1 \in \mathsf{Min}(\omega, G, d_2)$, a contradiction. Hence $|C| = |K|$.

Define a matrix $\widehat{\mathcal{H}}$ with both coordinates indexed by $\mathsf{conf}(X)$ such that for $c, c' \in \mathsf{conf}(X)$, where $c = (V_0, V_1, V_2, M)$ and $c' = (V'_0, V'_1, V'_2, M')$:

$$\widehat{\mathcal{H}}[c, c'] = \begin{cases} 1 & \text{if } V_0 = V'_2,\ V_2 = V'_0, \text{ and } M \cup M' \text{ is a simple cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

Notice that if we sort the indices of $\widehat{\mathcal{H}}$ by the partitions (V_0, V_1, V_2), then $\widehat{\mathcal{H}}$ can be seen as a block diagonal matrix with one block for each partition, and this block is a compatibility matrix on V_1. That is,

$$\widehat{\mathcal{H}} = \bigoplus_{V_0 \uplus V_1 \uplus V_2 = X} \mathcal{H}_{V_1},$$

where $\bigoplus$ denotes the operator of combining several matrices into a single block diagonal matrix. By Theorem 18, the rank over $\mathbb{F}_2$ of each of these blocks is bounded by $2^{|X|/2-1}$, hence the rank over $\mathbb{F}_2$ of $\widehat{\mathcal{H}}$ is bounded by $2^{|X|/2-1} \cdot 3^{|X|} \leqslant 2^{\mathcal{O}(|X|)}$.

Next, we claim that the set of rows of $\widehat{\mathcal{H}}$ corresponding to the configurations of C is linearly independent over $\mathbb{F}_2$. Assume not, hence there is a nonempty set of configurations $D \subseteq C$ such that

$$\sum_{d\in D} \widehat{\mathcal{H}}[d, \cdot] = \mathbf{0},$$

where $\mathbf{0}$ is the all-zero vector (all computations are performed in $\mathbb{F}_2$). For each $d \in D$ there is some $S_d \in K$ such that $d = c(S_d)$. Let d_{max} be a configuration of D for which $\omega(S_{d_{\mathsf{max}}})$ is the largest possible. Since $d_{\mathsf{max}} \in C$, we have that $\mathsf{Min}(\omega, G, c) = \{S_{d_{\mathsf{max}}}\}$ for some $c \in \mathsf{conf}(X)$ and hence $\widehat{\mathcal{H}}[d_{\mathsf{max}}, c] = 1$. However, as $\sum_{d\in D} \widehat{\mathcal{H}}[d, \cdot] = \mathbf{0}$, there must be another $d' \in D$, $d' \neq d_{\mathsf{max}}$, such that also $\widehat{\mathcal{H}}[d', c] = 1$. This means that d' is compliant with c, which implies that $\omega(S_{d'}) > \omega(S_{d_{\mathsf{max}}})$ by $\mathsf{Min}(\omega, G, c) = \{S_{d_{\mathsf{max}}}\}$. This contradicts the maximality of $\omega(S_{d_{\mathsf{max}}})$.

We conclude that the set of rows of $\widehat{\mathcal{H}}$ corresponding to C are indeed linearly independent over $\mathbb{F}_2$. Therefore, $|K| = |C|$ is upper bounded by the rank of $\widehat{\mathcal{H}}$ over $\mathbb{F}_2$, which is at most $2^{\mathcal{O}(|X|)}$. ◀

3.3 Hamiltonian cycles in general graphs using $\mathcal{O}(n)$ random bits

We now use the tools prepared so far to prove Theorem 2. The goal is to isolate all Hamiltonian cycles in an undirected graph $G = (V, E)$ using $\mathcal{O}(n)$ random bits, where n is the vertex count. First we give the isolation procedure. Then we analyze the probability of isolating all Hamiltonian cycles using configurations, compliant partial solutions, and the rank-based approach (through Theorem 19). Throughout the subsection we assume without loss of generality that $\log n$ is an integer.

As usual with isolation schemes, we assume that the vertex set of the considered graph G is $V = [n]$. We will apply induction on specific subgraphs of G called *intervals*.

▶ **Definition 20** (Interval of G). *For integers $1 \leqslant s \leqslant t \leqslant n$ and $1 \leqslant s' \leqslant t' \leqslant n$, the* interval *$G\langle s,t,s',t'\rangle$ is the graph (V', E'), where*

$$V' \coloneqq \{s,\dots,t\} \cup \{s',\dots,t'\} \qquad \text{and} \qquad E' \coloneqq \{uv \colon u \in \{s,\dots,t\}, v \in \{s',\dots,t'\}, uv \in E\}.$$

By $V\langle s,t,s',t'\rangle$ we denote the vertex set V' of the interval $G\langle s,t,s',t'\rangle$.

Note that $G\langle s,t,s,t\rangle$ is just the subgraph of G induced by $\{s,\dots,t\}$. On the other hand, if $\{s,\dots,t\} \cap \{s',\dots,t'\} = \emptyset$, then $G\langle s,t,s',t'\rangle$ is a bipartite graph, with $\{s,\dots,t\}$ and $\{s',\dots,t'\}$ being the sides of the bipartition.

Isolation scheme. We first present the isolation scheme. Let $\mathsf{id} \colon E(G) \to \{1,\dots,|E(G)|\}$ be any bijection that assigns to each edge $e \in E(G)$ its unique *identifier* $\mathsf{id}(e)$. Let C be some large enough constant, to be chosen later. Then independently at random sample $1 + \log n$ primes $p_0, p_1, \dots, p_{\log n}$ so that p_i is sampled uniformly among primes in the range $\{1,\dots,M_i\}$, where $M_i \coloneqq 2^{C(\log n + 2^i)}$. Note that choosing each p_i requires $C(\log n + 2^i)$ random bits, hence we have used $\mathcal{O}(n)$ random bits in total.

Next, we inductively define weights functions $\omega_0, \dots, \omega_{\log n}$ on $E(G)$ as follows:

- Set $\omega_0(e) \coloneqq 2^{\mathsf{id}(e)} \bmod p_0$ for all $e \in E(G)$.
- For each $e \in E(G)$ and $i = 1, \dots, \log n$, set

$$\omega_i(e) \coloneqq M_{i-1} n \cdot \omega_{i-1}(e) + \left(2^{\mathsf{id}(e)} \bmod p_i\right).$$

Let $\omega \coloneqq \omega_{\log n}$ and observe that ω assigns weights bounded by $2^{\mathcal{O}(n)}$, as required.

Analysis. We will prove the following statement for all $0 \leqslant i \leqslant \log n$ using induction on i.

Induction hypothesis

With probability at least $\left(1 - \frac{1}{n^2}\right)^{i+1}$, for all intervals $G\langle s,t,s',t'\rangle$ s.t. $t - s \leqslant 2^i$ and $t' - s' \leqslant 2^i$ and for each configuration $c \in \mathsf{conf}(V\langle s,t,s',t'\rangle)$, there is at most one minimum weight (w.r.t. ω_i) compliant partial solution, i.e. $|\mathsf{Min}(\omega_i, G\langle s,t,s',t'\rangle, c)| \leqslant 1$.

For $i = \log n$, the induction hypothesis gives us that for the complete interval $G = G\langle 1,1,n,n\rangle$ and for the configuration $c = (\emptyset, \emptyset, V(G), \emptyset)$, there is at most one minimum weight compliant partial solution w.r.t. ω. In other words, w.r.t. ω there is at most one minimum weight Hamiltonian cycle in G. This happens with probability at least $\left(1 - \frac{1}{n^2}\right)^{\log n + 1} \geqslant 1 - \frac{1}{n}$. So it remains to perform the induction.

Base step. For $i = 0$, we have $t - s \leqslant 1$ and $t' - s' \leqslant 1$. Hence each such interval $G\langle s, t, s', t'\rangle$ has at most 4 edges. Let

$$Y := \bigcup_{\substack{t-s\leqslant 1\\ t'-s'\leqslant 1}} 2^{E(G\langle s,t,s',t'\rangle)}$$

and for each $S \in Y$, let

$$x_S := \sum_{e\in S} 2^{\mathsf{id}(e)}.$$

Observe that since the identifiers assigned to the edges are unique, the numbers x_S are also pairwise different. Also, note that $|Y| \leqslant 16n^2$ as there are at most n^2 intervals considered, and for each of them there are at most 16 possible subsets of the at most four edges. Recall that $M_0 = 2^{C(\log n+1)}$ and p_0 is drawn uniformly at random among the primes in the range $\{1, \ldots, M_0\}$. Therefore, from Lemma 11 we can conclude that with probability at least

$$\left(1 - \frac{(n^2+1)(16n^2)^2}{2^{(C/2)(\log n+1)}}\right) \geqslant \left(1 - \frac{1}{n^2}\right)$$

all the numbers $\{x_S \colon S \in Y\}$ have pairwise different remainders modulo p_0; here the last inequality holds for a large enough constant C. Since $\omega_0(S) \equiv x_S \bmod p_0$, this means that with probability at least $\left(1 - \frac{1}{n^2}\right)$, all $S \in Y$ receive pairwise different weights with respect to ω_0. Therefore, the induction hypothesis is true for $i = 0$.

Induction step. Assume the induction hypothesis is true for all intervals $G\langle s, t, s', t'\rangle$ such that $t - s \leqslant 2^{i-1}$ and $t' - s' \leqslant 2^{i-1}$. Let

$$Y' := \bigcup_{\substack{t-s\leqslant 2^{i-1}\\ t'-s'\leqslant 2^{i-1}}} \quad \bigcup_{c\in\mathsf{conf}(V\langle s,t,s',t'\rangle)} \mathsf{Min}(\omega_{i-1}, G\langle s,t,s',t'\rangle, c)$$

be the set of all the minimal partial solutions for those intervals. Further, let

$$Y := \{S_1 \cup S_2 \cup S_3 \cup S_4 \colon S_1, S_2, S_3, S_4 \in Y'\}$$

be the set containing all combinations of four such partial solutions. The strategy is as follows. We first prove in Claim 21 that any relevant minimum weight compliant partial solution should be in Y. Then Claim 22 says that with hight probability, all partial solutions $S \in Y$ have pairwise different weights with respect to ω_i. Hence, proving these two claims will be sufficient to make the induction hypothesis go through.

▷ Claim 21. Let $1 \leqslant a \leqslant b \leqslant n$ and $1 \leqslant a' \leqslant b' \leqslant n$ be such that $b - a \leqslant 2^i$ and $b' - a' \leqslant 2^i$, and let $c \in \mathsf{conf}(a, b, a', b')$. Then $\mathsf{Min}(\omega_i, G\langle a, b, a', b'\rangle, c) \subseteq Y$.

Proof. Take any $S \in \mathsf{Min}(\omega_i, G\langle a, b, a', b'\rangle, c)$. Let

$$r = \lceil (a+b)/2 \rceil \qquad \text{and} \qquad r' = \lceil (a'+b')/2 \rceil$$

and let us select

$$\begin{aligned} S_1 &\subseteq E(G\langle a, r-1, a', r'-1\rangle), & S_2 &\subseteq E(G\langle a, r-1, r', b'\rangle),\\ S_3 &\subseteq E(G\langle r, b, a', r'-1\rangle), & S_4 &\subseteq E(G\langle r, b, r', b'\rangle) \end{aligned}$$

so that S_1, S_2, S_3, S_4 are disjoint and $S = S_1 \cup S_2 \cup S_3 \cup S_4$. See Figure 3 for an example.

We argue that $S_1 \in \mathsf{Min}(\omega_{i-1}, G\langle a, r-1, a', r'-1\rangle, c_1)$ for some $c_1 \in \mathsf{conf}(V\langle a, r-1, a', r'-1\rangle)$. Let $c = (V_0, V_1, V_2, M)$. Since $S \cup M$ is a simple cycle that visits all vertices of $V\langle a, b, a', b'\rangle$, we see that $R \coloneqq S_2 \cup S_3 \cup S_4 \cup M$ is a partial solution in the graph $G\langle a, b, a', b'\rangle$ with the edges of M added. Letting $(V_0', V_1', V_2', M') \coloneqq c_{V\langle a, r-1, b, r-1\rangle}(R)$, it follows that S_1 is compliant with the configuration

$$c_1 \coloneqq (V_0' \setminus (V_2 \cap V\langle a, r-1, b, r-1\rangle)), V_1', V_2' \cup (V_2 \cap V\langle a, r-1, b, r-1\rangle), M').$$

Moreover, that $S \in \mathsf{Min}(\omega_i, G\langle a, b, a', b'\rangle, c)$ implies that $S_1 \in \mathsf{Min}(\omega_i, G\langle a, r-1, a', r'-1\rangle, c_1)$, for otherwise S_1 could be replaced in S with a smaller-weight partial solution S_1' that would be still compliant with c_1, and this would turn S into a smaller-weight partial solution $S' = S_1' \cup S_2 \cup S_3 \cup S_4$ that would be still compliant with c. Finally, by the construction of ω_i, $S_1 \in \mathsf{Min}(\omega_i, G\langle a, r-1, a', r'-1\rangle, c_1)$ entails $S_1 \in \mathsf{Min}(\omega_{i-1}, G\langle a, r-1, a', r'-1\rangle, c_1)$.

Therefore $S_1 \in Y'$. Analogously we argue that $S_2, S_3, S_4 \in Y'$, hence we conclude that $S \in Y$. ◁

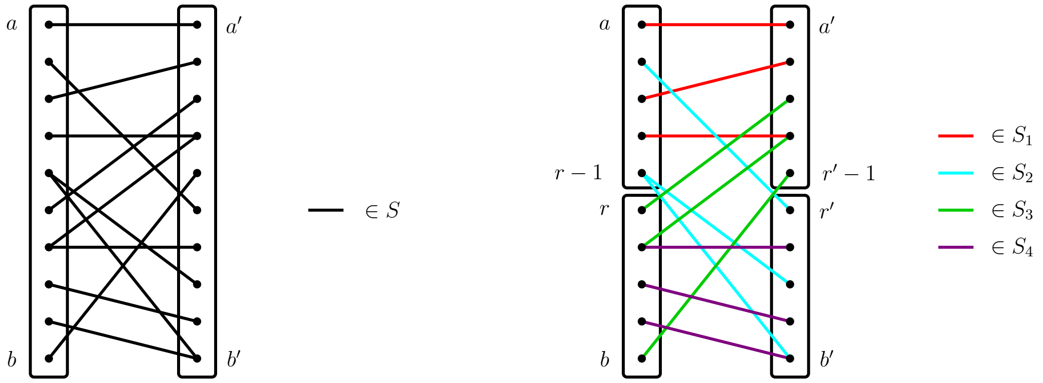


Figure 3 Example of splitting a partial solution $S \in E(G\langle a, b, a', b'\rangle)$ into four partial solutions S_1, S_2, S_3, S_4, where $S_1 \subseteq E(G\langle a, r-1, a', r'-1\rangle)$, $S_2 \subseteq E(G\langle a, r-1, r', b'\rangle)$, $S_3 \subseteq E(G\langle r, b, a', r'-1\rangle)$ and $S_4 \subseteq E(G\langle r, b, r', b'\rangle)$ with $r = \lceil (a+b)/2 \rceil$ and $r' = \lceil (a'+b')/2 \rceil$.

▷ Claim 22. The following event happens with probability at least $\left(1 - \frac{1}{n^2}\right)^{i+1}$: for all different $S, S' \in Y$, it holds that $\omega_i(S) \neq \omega_i(S')$.

Proof. For each $S \in Y$, let

$$x_S \coloneqq \sum_{e \in S} 2^{\mathsf{id}(e)}.$$

Observe that since identifiers assigned to the edges are unique, the numbers x_S are pairwise different. The induction hypothesis gives us that the following event A_{i-1} happens with probability at least $\left(1 - \frac{1}{n^2}\right)^{i}$: for all $1 \leqslant s \leqslant t \leqslant n$ and $1 \leqslant s' \leqslant t' \leqslant n'$ with $t - s \leqslant 2^{i-1}$ and $t' - s' \leqslant 2^{i-1}$, and all $c \in \mathsf{conf}(V\langle s, t, s', t'\rangle)$, we have $|\mathsf{Min}(\omega_{i-1}, G\langle s, t, s', t'\rangle, c)| \leqslant 1$. Assuming now that A_{i-1} indeed happens, by Theorem 19 we conclude that for every fixed choice of s, t, s', t' as above, we have

$$\left| \bigcup_{c \in \mathsf{conf}(V\langle s,t,s',t'\rangle)} \mathsf{Min}(\omega_{i-1}, G\langle s, t, s', t'\rangle, c) \right| \leqslant 2^{\mathcal{O}(2^{i-1})}.$$

Since there are at most n^4 choices of s, t, s', t', this implies that

$$|Y| \leqslant |Y'|^4 \leqslant 2^{\mathcal{O}(2^{i-1})} \cdot n^{16}.$$

Since $M_i = 2^{C(\log n + 2^i)}$ and p_i is drawn uniformly at random among the primes in the range $\{1, \ldots, M_i\}$, from Lemma 11 we can conclude that, for large enough C, with probability at least

$$\left(1 - \frac{(n^2+1)\left(n^{16}2^{\mathcal{O}(2^{i-1})}\right)^2}{2^{(C/2)(\log n + 2^i)}}\right) \cdot \left(1 - \frac{1}{n^2}\right)^i \geqslant \left(1 - \frac{1}{n^2}\right)^{i+1},$$

all the numbers $\{x_S : S \in Y\}$ have pairwise different remainders modulo p_i; here, the term $(1 - \frac{1}{n^2})^i$ corresponds to the probability that A_i happens. As a consequence, with the same probability we have that $\omega_i(S) \neq \omega_i(S')$ for all different $S, S' \in Y$. ◁

Now the induction step follows directly from combining Claim 21 with Claim 22.

4 Conclusion and directions for further research

In this paper we presented several isolation schemes for NP-complete problems, and we showed that analogues of decomposition-based methods such as Divide&Conquer can also be used to design more randomness-efficient isolation schemes. While we provide nearly matching lower bounds for all our results, at least as far as the number of random bits is concerned, we still leave open a number of interesting open questions:

1. Can we improve our isolation schemes to have weights that are only polynomial in n, while not increasing the number of used random bits? Note that in our approach, the use of large weights is crucial for the application of Lemma 11 that deals with interactions between different partial solutions in our isolation schemes.[3]
2. Can we shave off the log factors in the number of used random bits in our results? While some of the $\log n$ factors seem to be inherent in our ideas, there still might be a little room. For example, Melkebeek and Prakriya [36] presented an isolation scheme for reachability that uses $\mathcal{O}(\log^{1.5}(n))$-random bits. Perhaps with their ideas one can get the same guarantees for isolating Hamiltonian cycles in constant treewidth graphs.
3. Does the (even more) natural isolation scheme work as well? Many of our isolation schemes draw several random prime numbers and assign a weight that is obtained by concatenating the congruence class of the vertex/edge identifier with respect to the different primes. A more natural, but possibly harder to analyse, scheme would be to sample a single (larger) prime number and define the weights to be the congruence classes of the identifiers with respect to that single prime.
4. Our methods allowed us to derandomize polynomial-space algorithms for H-minor free graphs without significantly increase the running time. Can our methods be used to derandomize other algorithms likewise?

[3] In [8] a similar lemma was used to obtain isolation schemes with polynomial weights, but since the objects of the set family are not decomposed, the authors did not have this issue of interactions between different partial solutions.

References

1 Manindra Agrawal, Rohit Gurjar, and Thomas Thierauf. Impossibility of Derandomizing the Isolation Lemma for all Families. *Electron. Colloquium Comput. Complex.*, 27:98, 2020. URL: https://eccc.weizmann.ac.il/report/2020/098.

2 Manindra Agrawal, Thanh Minh Hoang, and Thomas Thierauf. The Polynomially Bounded Perfect Matching Problem Is in NC^2. In *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science*, pages 489–499, 2007. doi:10.1007/978-3-540-70918-3_42.

3 Rahul Arora, Ashu Gupta, Rohit Gurjar, and Raghunath Tewari. Derandomizing Isolation Lemma for $K_{3,3}$-free and K_5-free Bipartite Graphs. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*, pages 10:1–10:15, 2016. doi:10.4230/LIPIcs.STACS.2016.10.

4 Vikraman Arvind and Partha Mukhopadhyay. Derandomizing the isolation lemma and lower bounds for circuit size. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 276–289. Springer, 2008.

5 Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

6 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

7 Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comput. Theory*, 1(1):4:1–4:17, 2009. doi:10.1145/1490270.1490274.

8 Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-Optimal Unique Element Isolation with Applications to Perfect Matching and Related Problems. *SIAM J. Comput.*, 24(5):1036–1050, 1995. doi:10.1137/S0097539793250330.

9 Jiehua Chen, Wojciech Czerwiński, Yann Disser, Andreas Emil Feldmann, Danny Hermelin, Wojciech Nadara, Michał Pilipczuk, Marcin Pilipczuk, Manuel Sorge, Bartlomiej Wróblewski, and Anna Zych-Pawlewicz. Efficient fully dynamic elimination forests with applications to detecting long paths and cycles. *CoRR*, abs/2006.00571, 2020. To appear in the proceedings of SODA 2021. arXiv:2006.00571.

10 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity Checking Via Bases of Perfect Matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 150–159. IEEE, 2011. doi:10.1109/FOCS.2011.23.

12 Wojciech Czerwiński, Wojciech Nadara, and Marcin Pilipczuk. Improved Bounds for the Excluded-Minor Approximation of Treedepth. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 34:1–34:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.34.

13 Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory Comput. Syst.*, 47(3):737–757, 2010. doi:10.1007/s00224-009-9204-8.

14 Zdenek Dvořák, Archontia C. Giannopoulou, and Dimitrios M. Thilikos. Forbidden graphs for tree-depth. *Eur. J. Comb.*, 33(5):969–979, 2012. doi:10.1016/j.ejc.2011.09.014.

15 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019. arXiv:1904.01361.

16 Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. In *27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012*, pages 265–274. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.37.

17 Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. *SIAM Journal on Computing*, pages STOC16–218, 2019.

18 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

19 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a Sparse Table with $\mathcal{O}(1)$ Worst Case Access Time. *J. ACM*, 31(3):538–544, 1984. doi:10.1145/828.1884.

20 Martin Fürer and Huiwen Yu. Space saving by dynamic algebraization based on tree-depth. *Theory Comput. Syst.*, 61(2):283–304, 2017. doi:10.1007/s00224-017-9751-3.

21 Dima Grigoriev and Marek Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 166–172. IEEE Computer Society, 1987. doi:10.1109/SFCS.1987.56.

22 Chetan Gupta, Vimal Raj Sharma, and Raghunath Tewari. Efficient Isolation of Perfect Matching in $\mathcal{O}(\log n)$ Genus Bipartite Graphs. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

23 Falko Hegerfeld and Stefan Kratsch. Solving Connectivity Problems Parameterized by Treedepth in Single-Exponential Time and Polynomial Space. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020*, pages 29:1–29:16, 2020. doi:10.4230/LIPIcs.STACS.2020.29.

24 Bart M. P. Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *Theor. Comput. Sci.*, 795:520–539, 2019. doi:10.1016/j.tcs.2019.08.006.

25 Jason Li and Jesper Nederlof. Detecting Feedback Vertex Sets of Size k in $\mathcal{O}^\star(2.7^k)$ Time. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 971–989. SIAM, 2020. doi:10.1137/1.9781611975994.58.

26 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

27 Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Wegrzycki. Isolation schemes for problems on decomposable graphs. *CoRR*, abs/2105.01465, 2021. arXiv:2105.01465.

28 Jesper Nederlof, Michał Pilipczuk, Céline M. F. Swennenhuis, and Karol Węgrzycki. Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. In *46th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2020*, volume 12301 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2020. doi:10.1007/978-3-030-60440-0_3.

29 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity — Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

30 Michał Pilipczuk and Sebastian Siebertz. Polynomial bounds for centered colorings on proper minor-closed graph classes. In *30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1501–1520. SIAM, 2019. doi:10.1137/1.9781611975482.91.

31 Michał Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *ACM Trans. Comp. Theory*, 9(4):18:1–18:36, 2018. doi:10.1145/3154856.

32 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000. doi:10.1137/S0097539798339041.

33 Barkley Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63(1):211–232, 1941. URL: http://www.jstor.org/stable/2371291.

34 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in Quasi-NC. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 696–707, 2017. doi:10.1109/FOCS.2017.70.

35 Leslie G. Valiant and Vijay V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

36 Dieter van Melkebeek and Gautam Prakriya. Derandomizing Isolation in Space-Bounded Settings. *SIAM J. Comput.*, 48(3):979–1021, 2019. doi:10.1137/17M1130538.

37 Avi Wigderson. $\mathsf{NL}/\mathsf{poly} \subseteq \oplus\mathsf{L}/\mathsf{poly}$ (preliminary version). In *Proceedings of the Ninth Annual Structure in Complexity Theory Conference*, pages 59–62, 1994. doi:10.1109/SCT.1994.315817.

38 Ryan Williams. Finding paths of length k in $\mathcal{O}^\star(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

A Hamiltonian cycles in graphs of bounded treewidth

We will now use the same approach to give a proof of Theorem 3. More precisely, assume we are given a graph G of treewidth at most k. Our goal is to isolate the family of Hamiltonian cycles in G using $\mathcal{O}(k \log n + \log^2 n)$ random bits.

The proof follows the same structure as that of Theorem 2. We first describe the isolation scheme and then analyze the scheme using a tree decomposition $\mathbb{T} = (T, \beta)$ of G of width at most k. Note that the actual decomposition is not needed for the isolation procedure, and is only used as a tool in the analysis.

Isolation scheme. We first present the isolation scheme. As before, we assume that $V(G) = [n]$ and n is a power of 2. Let $\mathsf{id}\colon E(G) \to \{1, \ldots, |E(G)|\}$ be any bijection that assigns to each edge $e \in E(G)$ its unique *identifier* $\mathsf{id}(e)$. Let C be some large enough constant, to be chosen later. Then we independently sample $3 \log n$ primes $p_1, \ldots, p_{3 \log n}$ so that each p_i is sampled uniformly among all primes in the interval $\{1, \ldots, M\}$, where $M = 2^{C(k \log n)}$. Note that choosing each p_i requires $C(k + \log n)$ random bits, hence we have used $\mathcal{O}(k \log n + \log^2 n)$ random bits in total, as required.

Next, we inductively define weights functions $\omega_0, \ldots, \omega_{3 \log n}$ on $E(G)$ as follows:

- Set $\omega_0(e) := 0$ for all $e \in E(G)$.
- For each $e \in E(G)$ and $i = 1, \ldots, 3 \log n$, set

$$\omega_i(e) := Mn \cdot \omega_{i-1}(e) + \left(2^{\mathsf{id}(e)} \bmod p_i\right).$$

We let $\omega := \omega_{3 \log n}$ and we observe that ω assigns weights bounded by $2^{\mathcal{O}(k \log n + \log^2 n)}$.

Analysis. Let $\mathbb{T} = (T, \beta)$ be a tree decomposition of G of width at most k. It is well-known that T can be chosen so that it has at most n nodes. Further, let $\eta := E(G) \to V(T)$ be any function that assigns to each edge e of G any node x of T such that $e \subseteq \beta(x)$. In the sequel we will assume that η is injective. This can be achieved by adding, for each node $x \in V(T)$, $|\eta^{-1}(x)| - 1$ new nodes with the same bag and adjacent only to x, and appropriately distributing the images of edges of $\eta^{-1}(x)$ among the new nodes. Note that after this modification, the number of nodes of T is bounded by $\binom{k+1}{2} \cdot n \leqslant n^3$.

Compared to the proof of Theorem 2, instead of intervals we will use *segments* in the tree T underlying the tree decomposition $\mathbb{T}$. Recall that segments have been defined and discussed in Section 2. We first observe that there are only few segments.

▷ Claim 23. There are at most n^9 segments of T.

Proof. Note that a segment I in T can be uniquely determined by specifying the at most two vertices of ∂I and any vertex of $V(I) \setminus \partial I$, provided there exists any. Since T has at most n^3 nodes, there are at most n^9 choices for such a specification. $\lhd$

For a set of nodes $Z \subseteq V(T)$, we write $\beta(Z) \coloneqq \bigcup_{z \in Z} \beta(z)$. Further, for a segment I of T we consider the graph

$$G\langle I\rangle \coloneqq \big(\beta(V(I)), \eta^{-1}(V(I))\big)\,.$$

Usually when speaking about partial solutions in $G\langle I\rangle$, we consider their configurations on the vertex subset $\beta(\partial I)$. Note that $G\langle T\rangle = G$.

We proceed to the induction. We will prove the following statement for all $0 \leqslant i \leqslant \log n$.

Induction hypothesis

With probability at least $\left(1 - \frac{1}{n^2}\right)^i$, for all segments I of T of size at most 2^i and for each configuration $c \in \mathsf{conf}(\beta(\partial I))$, there is at most one minimum weight (w.r.t. ω_i) compliant partial solution in $G\langle I\rangle$, i.e. $|\mathsf{Min}(\omega_i, G\langle I\rangle, c)| \leqslant 1$.

Note that since $|V(T)| \leqslant n^3$, for $i = 3\log n$ the induction hypothesis gives that for $G\langle T\rangle = G$, there is at most one Hamiltonian cycle that has the minimum weight w.r.t. ω with probability at least $\left(1 - \frac{1}{n^2}\right)^{3\log n} \geqslant \left(1 - \frac{1}{n}\right)$.

Base step. For $i = 0$, we take segments of size at most 1, i.e. we prove the induction hypothesis for every segment I of T that has either one or two nodes. More precisely, we have to prove that (with suitably large probability), for every such segment I and configuration $c \in \mathsf{conf}(\beta(\partial I))$, we have $|\mathsf{Min}(\omega_0, G\langle I\rangle, c)| \leqslant 1$. Note that since I has at most two nodes and η is injective, the edge set $E(G\langle I\rangle)$ consists of at most two edges. Moreover, it cannot be that two different edge subsets $E_1, E_2 \subseteq E(G\langle I\rangle)$ are simultaneously compliant with the same configuration $c \in \mathsf{conf}(\beta(\partial I))$. It follows that sets $\mathsf{Min}(\omega_0, G\langle I\rangle, c)$ have sizes at most 1 always, so the induction hypothesis for $i = 0$ is true.

Induction step. Assume the induction hypothesis is true for all segments of size at most 2^{i-1}. Let

$$Y' \coloneqq \bigcup_{I\colon \text{segment of size } \leqslant 2^{i-1}} \ \bigcup_{c \in \mathsf{conf}(\beta(\partial I))} \mathsf{Min}(\omega_{i-1}, G\langle I\rangle, c).$$

be the set of all minimum weight partial solutions for segments of size at most 2^{i-1}. Further, let

$$Y \coloneqq \{\, S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \ :\ S_1, S_2, S_3, S_4, S_5 \in Y' \,\}$$

be the set comprising all combinations of five such partial solutions.

We first prove with Claim 24 that every relevant minimum weight compliant edge is contained in Y. Then Claim 25 says that with high probability, all $S \in Y$ receive pairwise different weights with respect to ω_i. The induction hypothesis will follow directly from combining these two claims.

$\rhd$ Claim 24. Let I be any segment of size at most 2^i and let $c \in \mathsf{conf}(\beta(\partial I))$. Then $\mathsf{Min}(\omega_i, G\langle I\rangle, c) \subseteq Y$.

Proof. Consider any $S \in \mathsf{Min}(\omega_i, G\langle I\rangle, c)$. By Lemma 13, there exist segments $I_1, \ldots, I_t$ ($t \leqslant 5$), each of size at most 2^{i-1}, such that $E(I)$ is the disjoint union of $E(I_1), \ldots, E(I_t)$. For each $j \in \{1, \ldots, t\}$ choose $S_j \in E(G\langle I_j\rangle)$ so that S is the disjoint union of $S_1, \ldots, S_t$. The same argument as that was used in the proof of Claim 21 shows that there exists $c_j \in \mathsf{conf}(\beta(\partial I_j))$ such that $S_j \in \mathsf{Min}(\omega_{i-1}, G\langle I_j\rangle, c_j)$. Hence $S_j \in Y'$ for all $j \in \{1, \ldots, t\}$, so it follows that $S \in Y$. ◁

▷ Claim 25. The probability of the following event is at least $\left(1 - \frac{1}{n^2}\right)^i$: for all different $S, S' \in Y$, it holds that $\omega_i(S) \neq \omega_i(S')$.

Proof. For each $S \in Y$ let us define

$$x_S = \sum_{e \in S} 2^{\mathsf{id}(e)}.$$

Observe that since the identifiers assigned to the edges are unique, the numbers x_S are pairwise different. By the induction hypothesis, the following event A_{i-1} happens with probability at least $\left(1 - \frac{1}{n^2}\right)^{i-1}$: for every segment I of size at most 2^{i-1} and each configuration $c \in \mathsf{conf}(\beta(\partial I))$, we have $|\mathsf{Min}(\omega_{i-1}, G\langle I\rangle, c)| \leqslant 1$. By Theorem 19 it follows that provided A_{i-1} happens, for every fixed segment I of size at most 2^{i-1} we have

$$\left| \bigcup_{c \in \mathsf{conf}(\beta(\partial I))} \mathsf{Min}(\omega_{i-1}, G\langle I\rangle, c) \right| \leqslant 2^{\mathcal{O}(|\beta(\partial I)|)} \leqslant 2^{\mathcal{O}(k)}.$$

By Claim 23 there are at most n^9 different segments, hence this implies that

$$|Y| \leqslant |Y'|^5 \leqslant 2^{\mathcal{O}(k)} \cdot n^{45}.$$

Recall now that $M = 2^{C(k + \log n)}$ and p_i is drawn uniformly at random among the primes in the range $\{1, \ldots, M\}$. Hence, from Lemma 11 we can conclude that, for large enough C, with probability at least

$$\left(1 - \frac{(n^2+1)\left(2^{\mathcal{O}(k)} \cdot n^{45}\right)^2}{2^{(C/2)(k + \log n)}}\right) \cdot \left(1 - \frac{1}{n^2}\right)^{i-1} \geqslant \left(1 - \frac{1}{n^2}\right)^i,$$

all the numbers in $\{x_S \colon S \in Y\}$ have pairwise different remainders modulo p_i. Here, the factor $(1 - \frac{1}{n^2})^{i-1}$ corresponds to the probability that A_{i-1} happens. As a consequence, with the same probability for all different $S, S' \in Y$ we have $\omega_i(S) \neq \omega_i(S')$. ◁

The induction step now follows directly from combining Claims 24 and 25.

Oritatami Systems Assemble Shapes No Less Complex Than Tile Assembly Model (ATAM)

Daria Pchelina (Дарья С Пчелина)
LIPN, Institut Galilée – Université Paris 13, France

Nicolas Schabanel
École Normale Supérieure de Lyon (LIP UMR5668 and IXXI, MC2), France

Shinnosuke Seki (関 新之助)
University of Electro-Communications, Tokyo, Japan

Guillaume Theyssier
Aix-Marseille Université, CNRS, I2M, Marseille, France

Abstract

Different models have been proposed to understand natural phenomena at the molecular scale from a computational point of view. Oritatami systems are a model of molecular co-transcriptional folding: the transcript (the "molecule") folds as it is synthesized according to a local energy optimisation process, in a similar way to how actual biomolecules such as RNA fold into complex shapes and functions. We introduce a new model, called *turedo*, which is a self-avoiding Turing machine on the plane that evolves by marking visited positions and that can only move to unmarked positions. Any oritatami can be seen as a particular turedo. We show that any turedo with lookup radius 1 can conversely be simulated by an oritatami, using a universal bead type set. Our notion of simulation is strong enough to preserve the geometrical and dynamical features of these models up to a constant spatio-temporal rescaling (as in intrinsic simulation). As a consequence, turedo can be used as a readable oritatami "higher-level" programming language to build readily oritatami "smart robots", using our explicit simulation result as a compiler.

As an application of our simulation result, we prove two new complexity results on the (infinite) limit configurations of oritatami systems (and radius-1 turedos), assembled from a finite seed configuration. First, we show that such limit configurations can embed any recursively enumerable set, and are thus exactly as complex as aTAM limit configurations. Second, we characterize the possible densities of occupied positions in such limit configurations: they are exactly the Π_2-computable numbers between 0 and 1. We also show that all such limit densities can be produced by one single oritatami system, just by changing the finite seed configuration.

None of these results is implied by previous constructions of oritatami embedding tag systems or 1D cellular automata, which produce only computable limit configurations with constrained density.

2012 ACM Subject Classification Computer systems organization → Molecular computing; Computing methodologies → Molecular simulation; Applied computing → Molecular structural biology; Theory of computation → Computability; Theory of computation → Complexity theory and logic

Keywords and phrases Molecular Self-assembly, Co-transcriptional folding, Intrinsic simulation, Arithmetical hierarchy of real numbers, 2D Turing machines, Computability

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.51

Supplementary Material
Software (Radius-1 Turedo to Delay-3 Oritatami compiler):
`https://hub.darcs.net/turedo2oritatami/turedo2oritatami/` [23]
Software (Oritatami simulator): `http://perso.ens-lyon.fr/nicolas.schabanel/OSsimulator` [22]

Funding *Nicolas Schabanel*: CNRS MITI MoPrExProgMol; IXXI Molécal; CNRS MITI AMARP; CNRS MITI DNASALGO; CNRS INS2I Algadène.
Shinnosuke Seki (関 新之助): JSPS KAKENHI Grant-in-Aids for Scientific Research (B) No. 20H04141 and (C) No. 20K11672.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 51; pp. 51:1–51:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A major trend of natural computing is the study of computational models inspired by molecular biology that are both theoretically rich and realistic enough to allow *in-vitro* implementations. Oritatami systems were introduced in [9, 10] to investigate the computational power of molecular co-transcriptional folding, in which an RNA sequence (transcript) folds upon itself into an intricate structure while being synthesized (transcribed). This phenomenon has proven programmable *in-vitro* by [12], in which Geary, Rothemund, and Andersen demonstrated how to encode a rectangular tile-like structure in a transcript and its folding pathway so that this transcript folds cotranscriptionally along the pathway into the encoded structure. This *RNA Origami* architecture has recently been highly automated by their software ROAD (RNA Origami Automated Design) [8]. ROAD extends the scale and functional diversity of RNA scaffolds, and is thus a promising direction for the design of RNA-based computation. DNA tile self-assembly did rely on the cellular automata theory to build up the abstract Tile Assembly model (aTAM) [24] which in turn allowed to develop experimental settings simple enough to be implement in vitro, such as the Sierpinski triangle [21]. On the opposite, RNA origami was born first *in-vitro* and the oritatami system was created [11] to answer the lack of theoretical framework to design computations for cotranscriptional-based assembly systems. In this paper, we introduce the *turedo* model, implementable in oritatami, which, as opposed to oritatami, is simple enough to program, to wish for a design equivalent to the Sierpinski triangle experiment for cotranscription-based *in-vitro* systems.

An oritatami system consists of a “molecule” (the *transcript*) made of “beads” that attract each other. The molecule grows by one bead per step and, at each step, the δ most recently produced beads are free to move around to look for the position that maximizes the number of bonds they can make with each other (hence the folding is co-transcriptional). This process ends up self-assembling a shape incrementally. It is known from [11, 20] that oritatami systems are Turing universal. They can also build arbitrary shapes [4] modulo a small universal constant upscaling, as well as specific fractals [17]. However, oritatami systems remain notably challenging to design. Indeed, the only shapes that can be built by [11, 20] are space-time diagrams of cyclic tag-systems or 1D cellular automata; and [4] requires to hardcode the whole shape in the transcript. The new computational model introduced in this article (*turedo*) not only abstracts away the technical details of attraction rules and bead sequence of oritatami, but embraces the geometrical aspects of them, as opposed to the simulation of classical one-dimensional computational models. We demonstrate that turedos can be simulated up to upscaling by oritatami systems. Our simulation allows thus to take full advantage of turedo computations in building shapes, and can be used as a compiler to design powerful oritatami systems as demonstrated below.

Oritatami systems and Turedos. The classical model of Turing machines has already been considered in other settings than the one dimensional bi-infinite tape, in particular in higher dimensions [1]. A popular class of Turing machines in $\mathbb{Z}^2$ is that of turmites [16], which are free to move on the plane but do it by just looking at their current internal state and the tape content at their current position. In this paper, we introduce a somewhat orthogonal class of Turing machines on the plane, that we call *turedos*[1], which can look at the tape content around their position to decide their move (like in [1]), but are constrained to move only in a self-avoiding way.

[1] Inspired by the nicely coined terminology for turmites, as a reference to *toredo navalis* (shipworms) that would only grow self-avoiding tunnels in wood if they were infinite.

Both our models (oritatami and turedos) have two strong constraints: they are sequential and self-avoiding (*i.e.* each position of the plane can only be visited once and becomes an obstruction for future moves). They can be seen as the sequential counterpart of aTAM model of self-assembly [19, 5] or freezing cellular automata [13, 2, 18]. But they are not just finite state automata growing a self-avoiding path in a regular way. Their computational power is in their ability to make moves depending on the configuration of neighboring positions.

Our main result is that oritatami can simulate turedos of lookup radius 1. Our notion of simulation is strong enough to preserve the geometrical and dynamical features of these models up to a constant spatio-temporal rescaling: the oritatami reproduces the whole dynamics of the turedo using macro-cells and a constant spatio-temporal rescaling. This definition is similar to intrinsic simulations developed for cellular automata [3] or self-assembly tilings [5]. Theorem 1.1 is proved in section 3.

▶ **Theorem 1.1** (Main result 1). *There is a universal bead type set $\mathcal{B}$ such that for any turedo $\mathcal{T}$ of radius 1 with alphabet of size Q, there is a delay-3 oritatami system based on $\mathcal{B}$ with period $\Lambda = \Theta(Q^6 \log Q)$ which simulates intrinsically $\mathcal{T}$at space-scale $\Theta(Q^3\sqrt{\log Q})$ and time-scale Λ.*

Complexity of limit configurations. The Turing universality results in [11, 20] induce undecidability results of the form: given an oritatami, a seed and a position, determining whether the position will be visited is undecidable. However, these embeddings are such that the obtained limit configurations are always computable because the space-time of the simulated tag system (or cellular automaton) computation is progressively constructed in a predictable way in a fixed region of oritatami's space. Precisely, in any limit configuration c^∞ obtained this way, the map $z \mapsto c^\infty(z)$ is computable because there is a computable time bound $\tau(z)$ such that if position z is not visited after $\tau(z)$ steps of the run, then it will never be visited (see Lemma 4.1).

The first application of our simulation result is to prove that we can produce uncomputable limit configurations from finite seeds with oritatami (section 4). This implies that there are oritatami runs from finite seeds where there is no computable time bound $\tau(z)$ on the visit time of position z.

Results on uncomputable limit configurations were already obtained in the model of directed aTAM [15]. Nevertheless, the construction used takes full advantage of the massive parallelism allowed in the aTAM model and *cannot* be translated into the turedo settings. Our construction is actually simpler than that of [15] and shows that sequential self-avoiding models can organize information in the plane in such a way that some regions allow "uncomputable comebacks".

▶ **Theorem 1.2** (Main result 2). *There exists a fixed oritatami with delay 3 and a fixed finite seed σ such that the produced limit configuration c_σ^∞ is uncomputable as a map.*

The second application of our simulation result is about (upper) density of occupied positions in the limit configurations obtained from finite seeds. Density is a natural geometrical parameter to test the ability of our models to produce complex infinite self-avoiding paths from finite seeds. We show that such densities are exactly the Π_2-computable numbers between 0 and 1 (Theorem 5.3), where Π_2-computable means being the limsup of a computable sequence of rational numbers [25]. In particular turedos and oritatami can produce limit densities which are not recursively approximable (i.e. not the limit of any computable sequence of rational numbers). We actually show that the whole spectrum of density can be obtained in a single turedo by varying the seed (Theorem 5.3). Using our simulation framework, the following result is shown first for turedos and then for oritatami in Section 5.

▶ **Theorem 1.3** (Main result 3). *For any $\epsilon > 0$, there exists an oritatami of delay 3 such that for any Π_2-computable number $d \in [0, 1-\epsilon]$, there is a finite seed σ such that the limit configuration c_σ^∞ reached from it has density of occupied positions exactly d.*

Note that the densities that can be produced in the (directed) aTAM model or freezing cellular automata from finite initial configurations cannot be more complex (see Lemma 5.1).

The organization of the paper is as follows: we first present oritatami and turedo models and the notion of simulation (section 2); then, we establish our main simulation result (section 3) and its two applications (sections 4 and 5).

2 Definitions and Models

Oritatami systems. Oritatami systems are embedded in the triangular lattice $\mathbb{T} = (\mathbb{Z}^2, \sim)$, where $(x,y) \sim (u,v)$ if and only if $(u,v) \in \cup_{\epsilon=\pm1}\{(x+\epsilon,y),(x,y+\epsilon),(x+\epsilon,y+\epsilon)\}$. Every position (x,y) in $\mathbb{T}$ is mapped in the euclidean plane to $x \cdot \overrightarrow{\mathsf{e}} + y \cdot \overrightarrow{\mathsf{sw}}$ using the vector basis $\overrightarrow{\mathsf{e}} = (1,0)$ and $\overrightarrow{\mathsf{sw}} = \mathrm{RotateClockwise}(\overrightarrow{\mathsf{e}}, 120°) = (-\frac{1}{2}, -\frac{\sqrt{3}}{2})$. We will denote by $\overrightarrow{\mathsf{nw}}, \overrightarrow{\mathsf{ne}}, \overrightarrow{\mathsf{e}}, \overrightarrow{\mathsf{se}}, \overrightarrow{\mathsf{w}}, \overrightarrow{\mathsf{sw}}$ the six canonical unit vectors in $\mathbb{T}$. Let B be a finite set of *bead types*. A *configuration* c of a bead type sequence $p \in B^* \cup B^{\mathbb{N}}$ is a directed self-avoiding path $c_0c_1c_2\cdots$ in $\mathbb{T}$, where for all integer i, the vertex c_i of c is labeled by p_i and refers to the *position* in $\mathbb{T}$ of the $(i+1)$-th bead in the configuration. A *partial configuration* of p is a configuration of a prefix of p.

For any partial configuration c of some sequence p, an *elongation* of c by k beads (or *k-elongation*) is a partial configuration of p of length $|c|+k$ extending by k positions the self-avoiding path of c. We denote by $\mathcal{C}_p$ the set of all partial configurations of p (the index p will be omitted whenever it is clear from the context). We denote by $c^{\triangleright k}$ the set of all k-elongations of a partial configuration c of sequence p.

An *oritatami system* $\mathcal{O} = (p, \heartsuit, \delta)$ is composed of (1) a (possibly infinite) bead type sequence p, called the *transcript*, (2) an *attraction rule*, which is a symmetric relation $\heartsuit \subseteq B^2$, and (3) a parameter δ called the *delay*. $\mathcal{O}$ is said to be *periodic* if p is infinite and periodic. Periodicity ensures that the “program” p embedded in the oritatami system is finite (does not hardcode unbounded behavior) and at the same time allows arbitrarily long computation.

We say that two bead types a and b *attract* each other when $a \heartsuit b$. Furthermore, given a (partial) configuration c of a bead type sequence q, we say that there is a *bond* between two adjacent positions c_i and c_j of c in $\mathbb{T}$ if $q_i \heartsuit q_j$ and $|i-j|>1$. The *number of bonds* of configuration c of q is denoted by $H(c) = |\{(i,j) : c_i \sim c_j,\ j > i+1, \text{ and } q_i \heartsuit q_j\}|$.

Oritatami dynamics. The folding of an oritatami system is controlled by the delay δ. Informally, the configuration grows from a *seed configuration* (the input), one bead at a time. This new bead adopts the position(s) that maximize(s) the potential number of bonds the configuration can make when elongated by δ beads in total. This dynamics is *oblivious* as it keeps no memory of the previously preferred positions [11].

Formally, given an Oritatami system $\mathcal{O} = (p, \heartsuit, \delta)$ and a *seed configuration* σ of a *seed bead type sequence* s, we denote by $\mathcal{C}_{\sigma,p}$ the set of all partial configurations of the sequence $s \cdot p$ elongating the seed configuration σ. The considered *dynamics* $\mathscr{D} : 2^{\mathcal{C}_{\sigma,p}} \to 2^{\mathcal{C}_{\sigma,p}}$ maps every subset S of partial configurations of length ℓ elongating σ of the sequence $s \cdot p$ to the subset $\mathscr{D}(S)$ of partial configurations of length $\ell+1$ of $s \cdot p$ as follows:

$$\mathscr{D}(S) = \bigcup_{c \in S} \operatorname*{arg\,max}_{\gamma \in c^{\triangleright 1}} \left(\max_{\eta \in \gamma^{\triangleright(\delta-1)}} H(\eta) \right)$$

The possible configurations at time t of the oritatami system $\mathcal{O}$ are the elongations of the seed configuration σ by t beads in the set $\mathscr{D}^t(\{\sigma\})$.

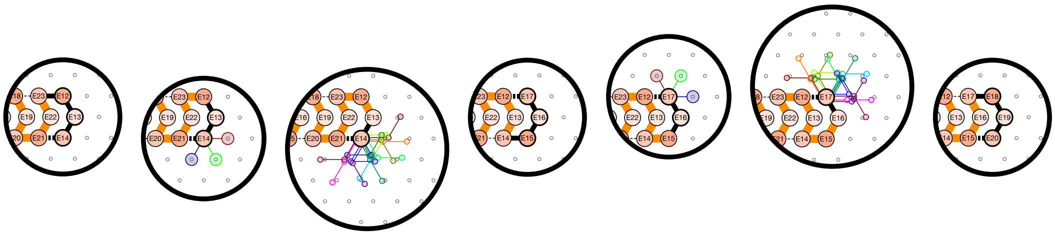


Figure 1 Oritatami model: From left to right, the growth from bead E12 to bead E18 of a self-supported oritatami glider with delay $\delta = 3$, transcript $p =$ E12...E23 and rule {E12 ❤ E17, E14 ❤ E21, E18 ❤ E23, E20 ❤ E15}. At each step, the set of nascent paths and maximizing the number of bonds is shown. The nascent beads are highlighted in bold black. The nascent paths are drawn in bold black until the last bond made and ends in colors when their tail is free to move (i.e., is not bounded by any bond).

We say that the Oritatami system is *deterministic* if at each time t, $\mathscr{D}^t(\{\sigma\})$ is either a singleton or the empty set. In this case, we denote by c^t the configuration at time t, such that: $c^0 = \sigma$ and $\mathscr{D}^t(\{\sigma\}) = \{c^t\}$ for all $t > 0$; we say that the partial configuration c^t *folds (co-transcriptionally) into* the partial configuration c^{t+1} deterministically. In this case, at time t, the $(t+1)$-th bead of p is placed at c^{t+1}, that is at the position that maximises the number of bonds that can be made in a δ-elongation of c^t. Figure 1 illustrates the folding steps of a delay-3 oritatami glider.

Turedos: Self-avoiding Turing Machines. A *turedo* is a Turing machine working on the plane with a lookup neighborhood (like in [1]), that can only move in a self-avoiding way. Turedos are embedded in the hexagonal lattice $\mathbb{H} = (\mathbb{Z}^2, \dot{\sim})$ whose 6 unit vectors are $N_H = \{\overrightarrow{\mathrm{N}} = (1,1), \overrightarrow{\mathrm{NE}} = (1,0), \overrightarrow{\mathrm{SE}} = (0,-1), \overrightarrow{\mathrm{S}} = (-1,-1), \overrightarrow{\mathrm{SW}} = (-1,0), \overrightarrow{\mathrm{NW}} = (0,1)\}$. Note that $\mathbb{H}$'s underlying grid is rotated by 30° with respect to $\mathbb{T}$'s . This choice is motivated by the main simulation result of the paper where macrocells in oritatami in our figures appear in the same orientation as the hexagonal cells in turedos. We denote by $B(r)$ the hexagonal ball of radius r centered on $(0,0)$, *i.e.* the set of positions in $\mathbb{Z}^2$ that can be written as a sum of at most r vectors from N_H. We also denote by $b(r)$ the size of $B(r)$, and $c_z(r) = (u \in B(r) \mapsto c(z+u))$ the restriction of a configuration c to the ball of radius r centered on z. Finally, we fix a universal blank symbol $\perp$ representing unoccupied positions.

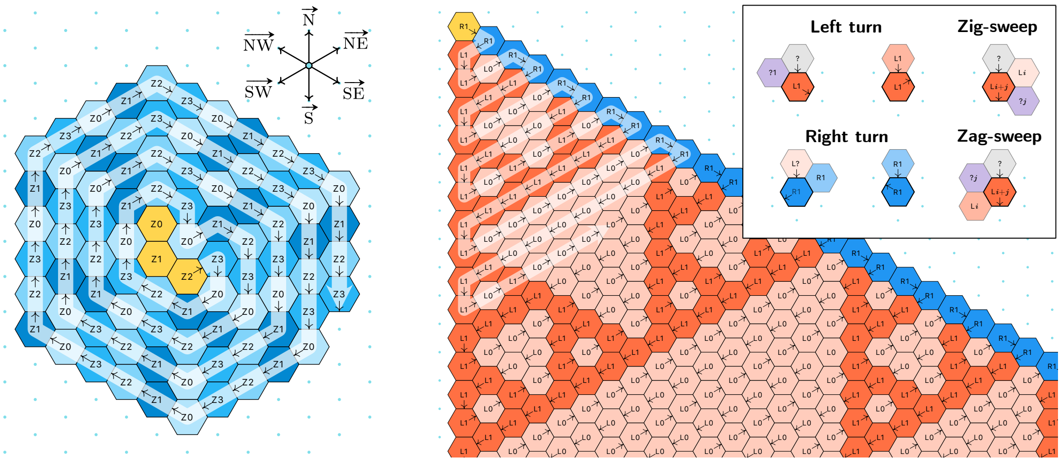
▶ **Definition 2.1.** *A* turedo *is defined by* $\mathcal{T} = (A, Q, q_0, r, \delta)$ *where* A *is the tape alphabet,* $\perp \in A$*,* Q *is the set of head states with initial state* $q_0 \in Q$*,* r *is the lookup radius,* $\delta : Q \times A^{B(r)} \to Q \times N_H \times A \smallsetminus \{\perp\}$ *is the local transition map.*

A tape configuration *is an element of* $A^{\mathbb{Z}^2}$*. A* global state *is an element of* $\mathcal{S}_\mathcal{T} = A^{\mathbb{Z}^2} \times \mathbb{Z}^2 \times Q$ *(tape configuration, position and state of the head). The turedo* $\mathcal{T}$ *induces a global map* $F_\mathcal{T} : \mathcal{S}_\mathcal{T} \to \mathcal{S}_\mathcal{T}$ *defined as follows:*

$$F_\mathcal{T}(c, z, q) = \begin{cases} (c, z, q) & \textit{if } c(z) \neq \perp \textit{ or } c(z+d) \neq \perp, \\ (c', z+d, q') & \textit{otherwise,} \end{cases}$$

where $(q', d, a) = \delta(q, c_z(r))$ *and image configuration* c' *is defined by:* $c'(z) = a$ *and* $c'(u) = c(u)$ *for* $u \neq z$*. When the first case occurs, we say that the machine is* blocked.

The key point of the above definition (which justifies the qualification of "self-avoiding") is that the only way tape configurations can be altered is by turning a blank symbol into a non-blank symbol, and therefore the head cannot go back to a previously visited position



(a) A 4-cyclic clockwise walker turedo. **(b)** A 3-states Sierpinski triangle. turedo

Figure 2 Two examples of radius-1 turedos. The seed configurations are displayed in yellow. The path followed by the turedo is highlighted in white. Empty (blank) positions are marked with a blue dot. **(a)** The turedo exits to the counterclockwise-most empty cell starting from its entry side. The states are just cycling in Z0, . . . , Z3. **(b)** The turedo uses three states L0, L1, R1 to perform a zigzag sweeping drawing the Sierpinski triangles pattern using the local rule given above.

(except when the machine is blocked in which case the global state is a fixed point). Positions holding a blank symbol are therefore seen as empty positions where the head can possibly move to. Two examples of radius-1 turedos are given in Figure 2.

Limit configuration and freezing time. Given an initial global state $s \in \mathcal{S}_{\mathcal{T}}$ for a turedo of global map $F_{\mathcal{T}}$, let us consider the sequence $(c^t, z_t, q_t) = F^t_{\mathcal{T}}(s)$ for $t \in \mathbb{N}$. By the self-avoiding property, it holds that for any $z \in \mathbb{Z}^2$ the sequence of symbols $(c^t(z))_{n\in\mathbb{N}}$ is ultimately constant, and, denoting its limit $c^\infty_s(z)$, we then have defined a tape configuration $c^\infty_s \in A^{\mathbb{Z}^2}$ which is called the *limit configuration* reached by F starting from s. Said differently, using the standard Cantor topology for tape configurations [14], we have that the sequence of configurations $(c^t)_t$ converges to c^∞_s. Moreover, we can associate to the system and the initial global state s, the *freezing time* map $\tau_s : \mathbb{Z}^2 \to \mathbb{N}$ such that $\tau_s(z)$ is the minimal t for which the tape content of cell z at time t is $c^\infty_s(z)$ (in particular, $\tau_s(z) = 0$ if $c^\infty_s(z) = \perp$).

Programming turedos. Thanks to the freedom allowed in their local maps, turedos are in general much easier to design than oritatami systems. The basic building block to design complex turedos is the zigzag sweeping movement which allows us to embed any 1D Turing machine/cellular automaton computation (see Fig. 2b). They can also be used as thick wires to transport information from one region to another.

Simulations. Any oritatami with delay δ can be seen as a particular turedo of radius $\delta + 1$: indeed, an oritatami transition is completely determined by the position in the sequence of beads, coded as a state of the turedo, and the local configuration in a ball of radius $\delta + 1$.

Our main result proven in the next section is a converse to this observation: any turedo of radius 1 can be simulated by an oritatami system of delay 3. The general idea is to reproduce the dynamics up to a linear spatio-temporal scale factor like in similar notions already considered for cellular automata or self-assembly tilings [3, 6, 2]. More precisely, each cell of the simulated system is represented by a macrocell in the simulator system, the macrocells form a linearly distorted hexagonal lattice, and a constant number of time steps is allowed for the simulator to reproduce one step of the simulated system. This notion of simulation is very strict and allows to relate properties of the limit configurations in the simulated system to the corresponding limit configuration in the simulator. This can be done without further hypothesis for computability of limit configurations, but can also be done for the density of non-blank states as soon as the simulation uses macrocells that are filled with the same high enough density.

3 Delay-3 oritatami systems simulate radius-1 Turedos

This section provides an overview of the design implying main Theorem 1.1. As for the 1D cellular automaton simulation in [20], our simulation proceeds in three phases: 1) reading the neighboring letters, 2) preparing for writing the new letter on the boundaries of the macrocell and 3) exiting to the computed next location. However, we must solve a significant number of new challenges to adapt to turedos. Turedos are free to move in every direction: the shape of the macrocells must then be isotropic. Furthermore, as the exit direction has to be deduced from the symbols read, the reading process must be non-blocking. Thus we cannot use the reading mechanism in [20], nor the writing flip-flap mechanism which would block any further return to a previously visited border; we cannot use its hardcoded exit mechanism either. Moreover, as we need to return to a random side after reading and writing on all sides, our oritatami system must be able to absorb up to 4 times the side length before exiting to the new macrocell and starting the next period of the transcript. It follows that we cannot park unused information on the boundary of the macrocell as in [20], but need to store information *inside* the macrocell to avoid increasing the macrocell side length uncontrollably. Similarly the speedbump module introduced in [20] must be adapted to fit inside a compact space.

To solve all those issues, we have developed new tools that we believe to be simple, powerful and generic enough to have their own interest. We also believe that some of them could serve as a guideline for a first biochemical implementation of computation using RNA co-transcription. Our current implementation `turedo2oritatami` uses 1735 bead types. Examples of radius-1 turedos compiled as oritatami as well as a fully functional `python` compiler can be downloaded from [23]: `https://hub.darcs.net/turedo2oritatami/turedo2oritatami/python`. The resulting `.os` files are to be run with the oritatami simulator by [22].

Bit-weight encoding of a Turedo. Consider a radius-1 turedo. First, we get rid of its internal state and orientation by encoding them in the symbols of the tape configuration. We then encode each symbol of the resulting tape alphabet $\mathcal{A}$ as a string of q bits where $q = \lceil \log_2 \#\mathcal{A} \rceil$. The blank symbol $\perp$ is encoded by the reserved word 0^q. Let $Q = 2^q$. In the following we assume that the neighboring cells of the current position are numbered in counterclockwise (CCW) order from 0 to 5 where 5 denotes the cell previously visited by the turedo. Our simulation assumes that the turedo transition function is a function $F : (2^q)^6 \to 2^q \times \{0, \dots, 4\}$, that reads the q bits $b_{i,0}, \dots, b_{i,q-1}$ encoding the symbol in the ith CCW neighboring cell for $i = 0..5$, and outputs the q bits of the symbol to be written

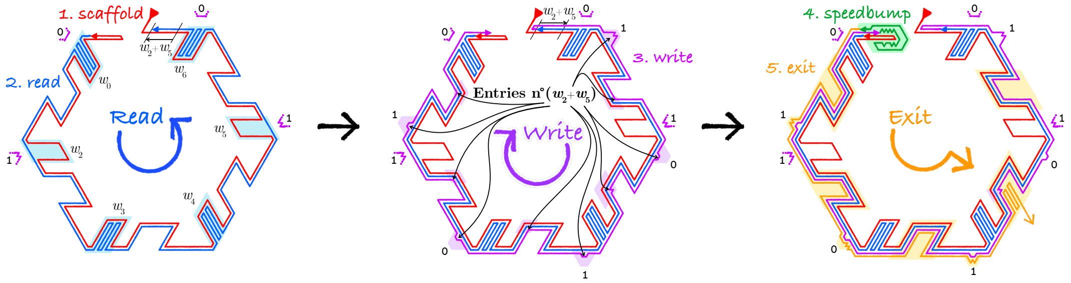


Figure 3 Principle of the macrocell operation. The shift of the reading layer at the end of its folding (and thus of the writing layer) is $\sum_{i:\text{bit read}_i=1} w_i = w_2 + w_5$.

and the CCW index of the next cell to go to.[2] Furthermore, we assume that F is encoded as a tuple $((w_{ij}), \Phi)$ such that $F((b_{ij})) = \Phi(\sum_{i,j} w_{ij} b_{ij})$ where the $6q$ bit-weights (w_{ij}) are non-negative integers. All transition function F can be encoded this way using the weights $w_{ij} = 2^{qi+j}$. We denote by $\mathcal{W} = \sum_{i,j} w_{ij}$ the sum of the weights of the bits. Encoded this way, the size of the transition table of F is exactly $\mathcal{W} + 1$ for every bit and the exit direction.

Principle of the macrocell operation. Fig. 3 presents a schematic overview of the key operations in the macrocell. The transcript consists of five parts:

1. the *scaffold* of the macrocell folds, on each side of the macrocell, in front of the position of each bit to be read, "read pockets" (in blue) of size equal to the weight given by the transition function to that bit; it also builds one "exit pocket" (in orange) per side;
2. the *read layer* folds counterclockwise and fills the read pockets (outlined in blue) when it senses a 0, and jumps over it when it senses a 1, pushing the transcript forward by a shift corresponding to the sum Δ of the sizes of the pockets sensing a 1 ($\Delta = w_2 + w_5$ in the figure);
3. the *write layer* contains all the transition tables of the simulated turedo, one for each bit to write on each side, and one for each exit-or-not decision on each side; this layer folds clockwise, and as it is translated forward by Δ, it folds the Δth entry of each transition table at the writing spots (in purple) that trigger the folding of the selected transition table entries. The shift Δ accumulated by the read layer allows then to write the output pattern on each side. It also places a "kicking bead" (in purple) in the exit pocket on the computed exit side and no-kicking beads in the other using the same shift-principle;
4. the *speedbump module* (outlined in green) absorbs the shift so that the next layer starts without any shift regardless of the values read by the read layer;
5. the *exit layer* folds counterclockwise, following the border until it hits the kick (outlined in yellow) and folds upon itself to the next macrocell.

Observe that the reading layer needs to "read" the bit from neighboring cells while still making room for the two next layers to fold between the reading layer and the neighboring cells. This explains why our oritatami systems has delay 3: it has to read through 3 layers.

This presentation was just an overview of the macrocell. The complete description of the macrocell is given in Fig. 4. We will now present some of the key tools used in our design.

[2] The case of attempting to exit towards the CCW neighboring cell n°5 from which the turedo came, is purposely ignored as it would unnecessarily complicate the construction.

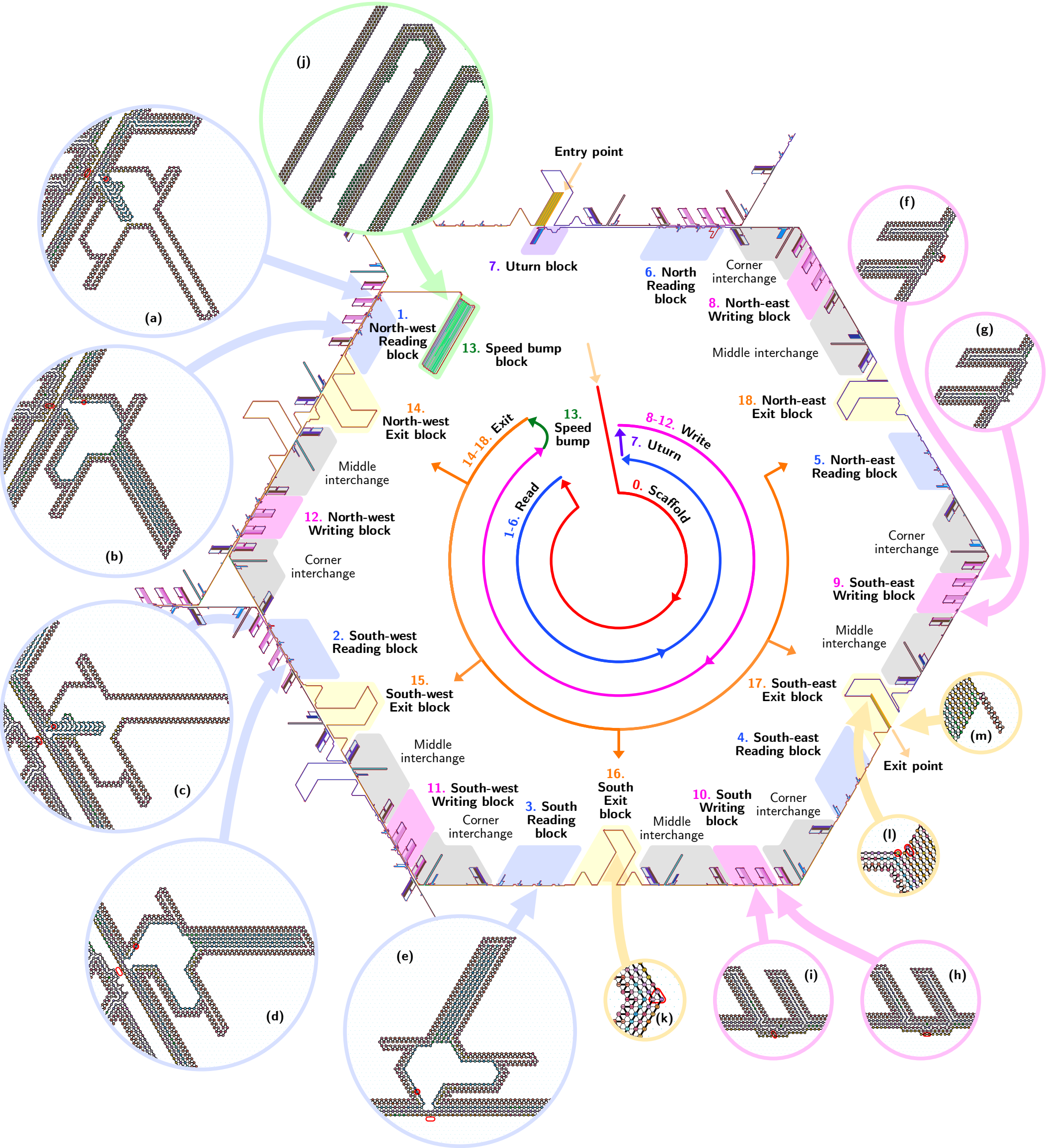


Figure 4 A macrocell for a turedo with $q = 3$ bits ($Q = 8$ tape symbols) together with the order in which layers and modules are used along its boundary as well as snapshots of important modules: (a)-(e) the read pocket in all possible situations: reading a 0/⊥ (fig. b, d and e) or a 1 (fig. a and c) from a neighboring cell (fig. a–d) (or not (fig. e)) and through its exit layer (fig. a and b) or directly from its write layer (fig. c and d) – (f)-(h) all possible situations for the write module: writing a 0 (fig. g and i) or a 1 (fig. f and h), through the exit layer (fig. h and i) or directly (fig. f and g) – (j) the shift-absorbing speedbump – (k) the exit layer folds along the exit pocket – (l)-(m) the write layer has placed the kicking bead ↻76 in the corner that detaches the exit layer from the pocket and concludes the folding by exiting to the SW. Zoom in for details

Folding meter and pockets. Our construction relies on two new simple and powerful tools:

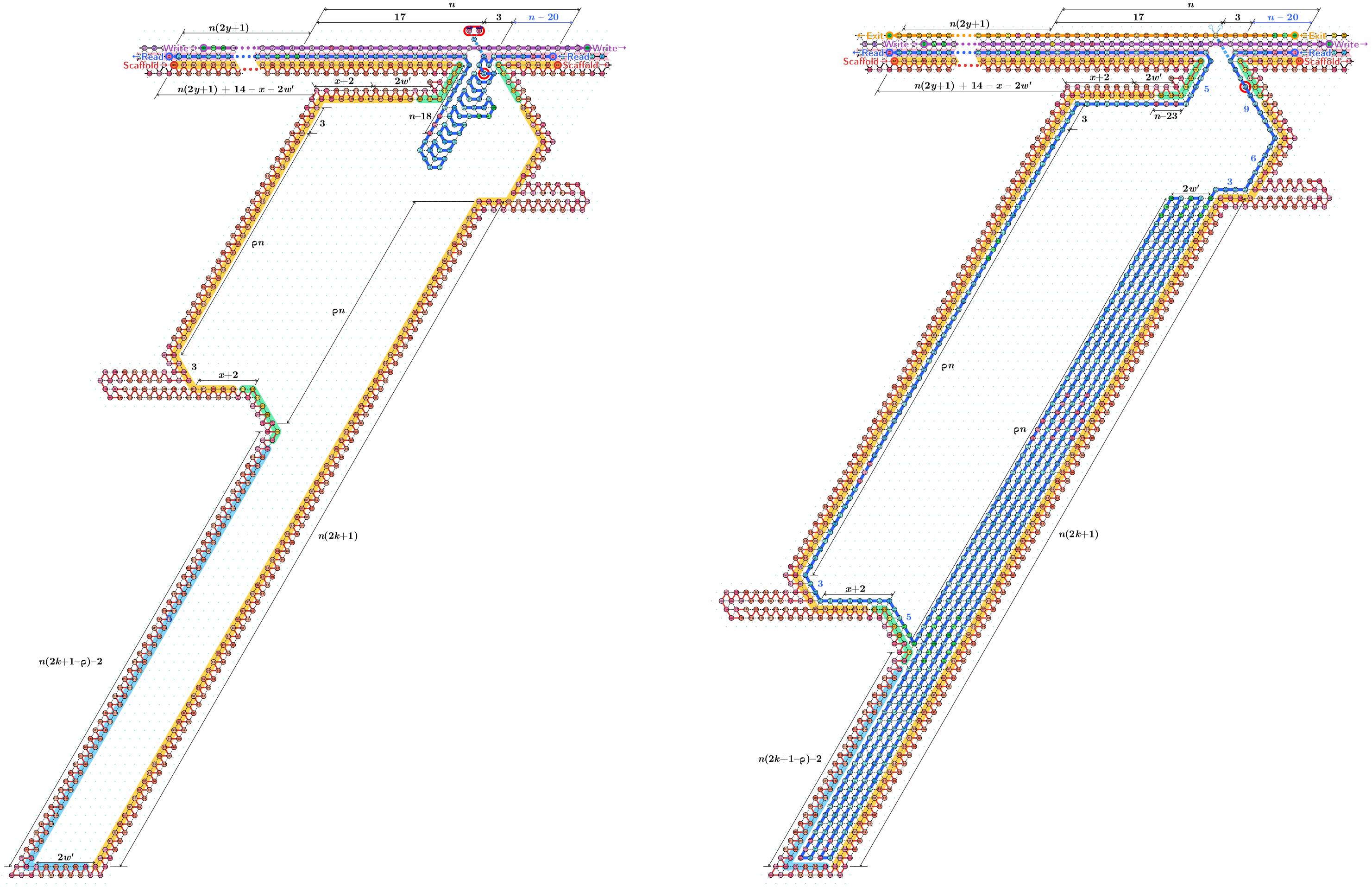
- a *folding meter* is a $4n$-periodic transcript whose period has 4 equally spaced articulation points, so that it can either: 1) follow a border if it is strongly attracted to it; 2) fold upon itself in a compact zig-zag form if the attraction to the border is weak; 3) reveal an hardcoded structure if the attraction to its surrounding is mild (see Appendix A).
- a *pocket* is a box which triggers the compact folding of a folding meter and which allows to hide a portion of it in a compact space. The entrance to such a pocket can be conditioned by the surrounding. For instance, the read folding meter enters a read pocket if and only if its reading head rq88 or rq36 is not attracted by two beads encoding a 1 in its neighborhood (see Fig. 4(a–e)), otherwise it folds into an hardcoded glider and exits the pocket rightaway.

Furthermore, several folding meters can be layered on top of each other in opposite directions as long as their periods match. Synchronizing and desynchronizing the two layers allow to trigger the various behaviors as well, by varying the strength of their bonds. For instance, the write layer folds into spikes encoding 0 or 1 when it passes over the read layer in Fig. 4(f–i) because its bonds are weaker with the read layer when the latter is desynchronized after having been sucked into the pitfalls that surround this area. The length n of the each segment is a priori arbitrary as long as $n \geqslant 6$. In the present design, n was set to 26 in order to accommodate all the desired configurations – the most demanding being the glider at the entrance of a read pocket when reading a 0, and the writing of 0 or 1 over the write module.

Read layer and pocket. Let us illustrate the folding meter/pocket mechanism with the read pocket used to induce a shift of $2n\kappa$ in the transcript every time it reads a 1 with bit-weight κ. The primary purpose of the read pocket is namely to read a bit (0/1) and to push forward the read layer by the amount equivalent to its capacity if the read layer reads a 1. The read layer folds from right to left. When the read layer reaches the entrance (see Fig. 5a), its "reading head", the bead rq88, "senses" whether there is a 1 written on the adjacent macrocell at this location. If there is a 1, encoded by the presence of the pair of beads ⊔|p62 and ⊔|p64 in Fig. 5a, then the reading head rq88 is momentarily attracted upwards (making two temporary bonds), which results in placing bead rq86 away from the border of the read pocket; the read layer gets then too far from the read pocket border to get attracted to it anymore, and folds into its natural hidden shape: a glider that will immediately escape from the read pocket (Fig. 5a). Otherwise, if there is a 0, encoded by the absence of these bead types at the expected location in Fig. 5b, the reading head rq88 gets attracted downwards inside the pocket by making one bond; the read layer pursues its course downwards along the border until it reaches the bottom of the pocket which does not attract it anymore; the read layer prefers then to fold upon itself into a switchback pattern, filling the pocket completely, until it reaches the other side of the pocket which attracts it again; the read layer follows this border until it exits the pocket. This results in a shift forward of the read layer by an amount corresponding to the pocket capacity $2n\kappa$ if and only if the bit written on the adjacent macrocell is 1.

Remark that this novel bit reading method, using a reading head, does not obstruct the way between adjacent cells unlike the method used in [20]; this allows the write and exit layers to pass and reach the exit at an arbitrary side. Note that this is the reason why our simulation uses delay 3.

Note finally that the interactions between the scaffold and the read layer are extremely simple: the only places where these interactions are carefully designed are at the entrance and at the end of the pocket (the three areas highlighted in green in Fig. 5), all the other



(a) Read pocket reading 1.

(b) Read pocket reading 0.

Figure 5 Read pocket. The *capacity* of a read pocket is defined to be the difference in length between the paths taken by the transcript upon reading 1 and upon reading 0, and it is determined by the parameters k, w, ρ with $\rho < 2k + 1$ as capacity $= 2n\left(w(2k+1) + \rho - 1 + \frac{(w+19).\mathsf{nextMultiple(of:}\ n)}{n}\right)$.

interactions are either “attract-them-all” (the areas highlighted in yellow) or “attract-none-of-them” (the areas highlighted in blue). This demonstrates the simplicity of the folding meter/pocket concept.

The read and write blocks. Fig. 6 shows in details the actual oritatami implementation of the read and write blocks and how write pockets of size equal to the size of the transition tables are used as interconnected vessels to place the correct entry of the table over each write module. The write module (responsible for writing 0 or 1 on the sides of the macrocell) and write pocket (responsible for hiding the unused entries of the transition table) are presented in details in Appendix B.

Layer interchange. Each layer is heavily interacting with its neighboring layers inside a macrocell. It follows that unwanted interferences may occur between facing layers from neighboring macrocells. For this purpose, we use three different variants of bead types in each layer: one for each half of each side (e.g. Read1 and Read2 for the read layer), plus one in the middle and the corners which interconnects one to the other and cancel the need for interactions between them (e.g., Read12 to switch between Read1 and Read2).

Setting up the exit block and computing the macrocell size. As the exit pocket needs to accommodate the remaining of the exit layer before it exits, it must have room to fold in a compact shape a folding meter of length up to four macrocell-sides long. Since the exit pocket belongs to the macrocell side, extending the exit pocket extends the macrocell side as well: we have thus to solve a fix point problem. Moreover, since a different amount of the exit layer will fold into each exit pocket depending on its location in the macrocell, we need a mechanism to make sure that, in all cases, the transcript will exit at the same position on every macrocell side, without interfering with the fix point resolution above. The latter problem is solved by using a pair of “loose ropes” of equal length, one on each side, “pulling” on the exit pocket to adapt its position to the macrocell side (see the two triangles of varying depth surrounding each of the five exit pockets in Fig. 4). Making the exit pocket deep enough allows to solve this fix point issue, which concludes the proof overview of Thm. 1.1.

4 Uncomputable Limit Configurations and Freezing Time

A configuration $c \in A^{\mathbb{Z}^2}$ is *computable* if there is a Turing machine which on input $z \in \mathbb{Z}^2$ computes $c(z)$. We are interested in the computability of limit configurations obtained from finite initial configurations (*i.e.* everywhere $\perp$ except on a finite region).

As said in the introduction, constructions of Turing universal oritatami systems known so far [20, 11] *do not* produce uncomputable limit configurations. The key reason is that they have a computable *escape direction*: a direction $u \in \mathbb{Z}^2$ and a computable non-decreasing function μ such that $\mu(t) \to \infty$ and for any $t \in \mathbb{N}$, the position z_t of the head after t steps verifies $u \cdot z_t \geqslant \mu(t)$ where “$\cdot$” denotes the scalar product (i.e. the head globally moves away along the direction u). Such a computable escape direction appears naturally in these simulations because they are fundamentally simulations of space-time of one-dimensional systems: they work by growing successive 1D finite configurations and stacking them along a direction u that corresponds to the time of the simulated system. The simulation never goes back to previously stacked layers simply because computing one step of the 1D system is performed using the last stacked 1D configuration only. More generally:

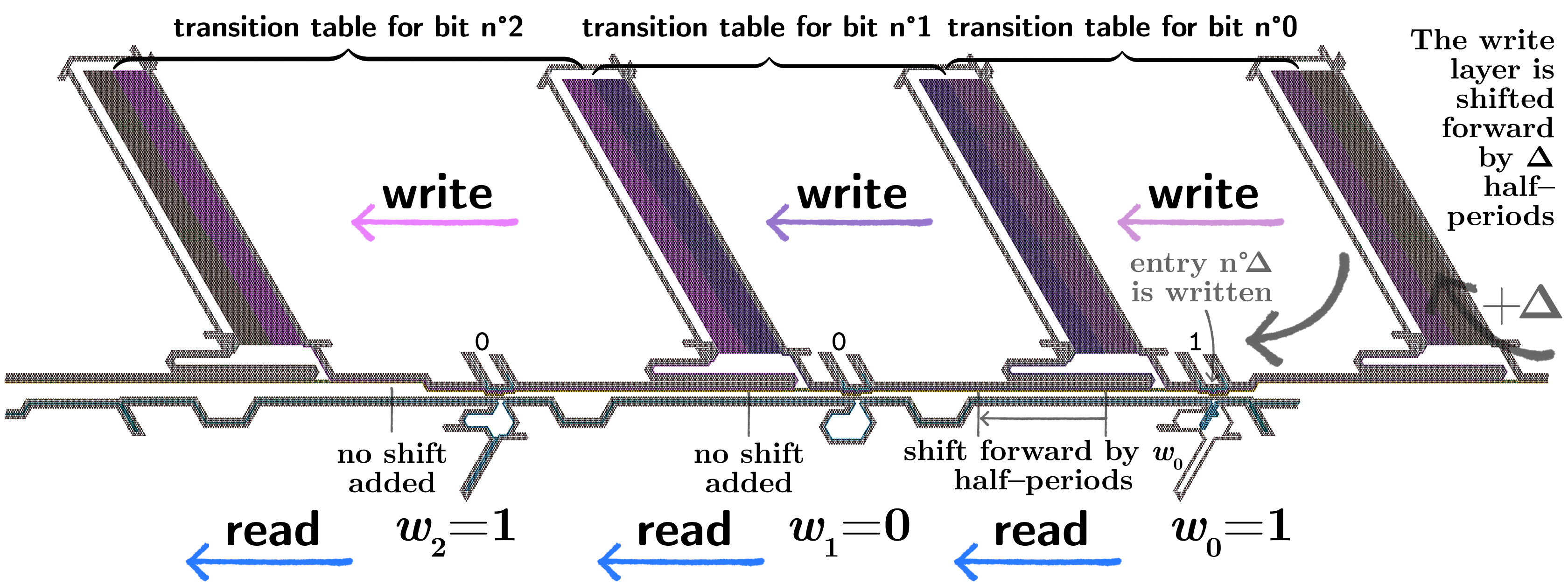
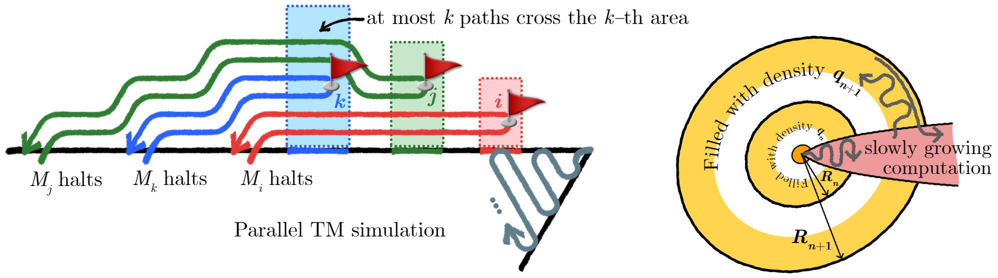

transition table for bit n°2
transition table for bit n°1
transition table for bit n°0
The write layer is shifted forward by Δ half–periods
write
write
write
entry n°Δ is written
+Δ
0
0
1
no shift added
no shift added
shift forward by w_0 half–periods
read
read
read
$w_2=1$
$w_1=0$
$w_0=1$

Figure 6 The read and write blocks: (bottom) when the read layer folds, it fills every read pocket facing a 0 on the neighboring macrocell, but skips the read pocket if it faces a 1, which increases the shift forward of the read layer by the weight of the corresponding bit; this yields a total additional shift of w_0 half-periods in the figure – (top) the transition tables are stored in the write layer transcript: one per half-period; the size of the write pockets is set to $\mathcal{W}$ half-periods to accomodate all the unused entries in the $(\mathcal{W}+1)$-long transition tables and the transition table is located in the write layer so that its first entry is aligned with the write module when the shift is 0; when the write layer starts to fold, it is shifted forward by $\Delta = \sum_{\text{bit read}(i,j)=1} w_{ij}$ half-periods, the total shift accumulated by its preceding read layer (each transition table is highlighted in a different color in the figure); this implies that the part of the write layer folding over each write module (highlighted in blue) is the one encoding the Δ-th entry of the transition table for each bit to write on the macrocell side as expected; this part will fold into a prescribed shape which will be read as 0 or a 1 by the read layer of its upcoming neighboring macrocell. This is an actual oritatami simulation. Zoom in for details


at most k paths cross the k–th area
k
j
i
M_j halts
M_k halts
M_i halts
Parallel TM simulation
Filled with density q_{n+1}
Filled with density q_n
R_n
R_{n+1}
slowly growing computation

(a) Sketch of the turedo building an uncomputable limit configuration.

(b) Sketch of the turedo building a limit configuration with an arbitrary density $d \in \Pi_2$.

Figure 7 Sketch of the two turedo constructions in sections 4 and 5.

▶ **Fact 4.1.** *For any turedo reaching limit configuration c_s^∞ from a finite global state s, the maps $z \mapsto c_s^\infty(z)$ and $z \mapsto \tau_s(z)$ are Turing-equivalent. Moreover, they are both computable if the dynamics admits a computable escape direction.*

In the next result, we construct a turedo that goes back uncomputably close to the origin uncomputably often in spite of following a self-avoiding trajectory. Precisely, we prove that turedos of radius 1 and therefore oritatami are powerful enough to embed any recursively enumerable set into their limit configurations reached from a finite initial configuration. As a consequence, both models produce uncomputable limit configurations.

▶ **Theorem 4.2.** *There exists a fixed turedo of radius* 1 *which, when started from a fixed global state s with a blank tape configuration, reaches an uncomputable limit configuration and therefore has an uncomputable freezing time map τ_s.*

Proof sketch. The basic idea, illustrated in Fig. 7a, is to build a turedo which runs a Turing machine simulation to test all Turing machines for halt in parallel and that, when it finds that some machine i has halted, interrupts momentarily its computation and goes to write a flag in a prefabricated area $p(i)$ located at a computable in i position (initially all areas $p(i)$ are empty). Areas of type $p(i)$ are progressively filled in some uncomputable and unknown order, but, at the limit, it holds that $p(i)$ contains a flag if and only if the machine i halts. Therefore the limit configuration is uncomputable because it can solve the halting problem when used as an oracle.

The key to implementing this idea is the layout of the paths to reach the areas $p(i)$: when we proceed as shown in Fig. 7a, no more than i paths will go across the area $p(i)$, i.e. the ones that correspond to the halting Turing machines j with $j < i$. As a zigzag of thickness $O(j)$ is enough for the turedo to reach area j, place a flag, and go back, then the flag in area $p(i)$ (if any) will never be placed higher than $O(i^2)$. It follows that these areas have quadratic size and their ground basis can be set up in advance by the turedo as it simulates the Turing machines in parallel (in particular, the turedo will start the simulation of machine i only after the ground basis of area $p(i)$ is set up). Of course, Figure 7a is a simplification and does not represent all movements of the turedo's head: in particular, when moving towards area $p(i)$, the turedo needs to carry on the information i and to bubble up the ground basis of each area crossed over along the way, and it cannot carry those in its state set. Using our simulation framework, Theorem 1.2 follows from Theorems 1.1 and 4.2.

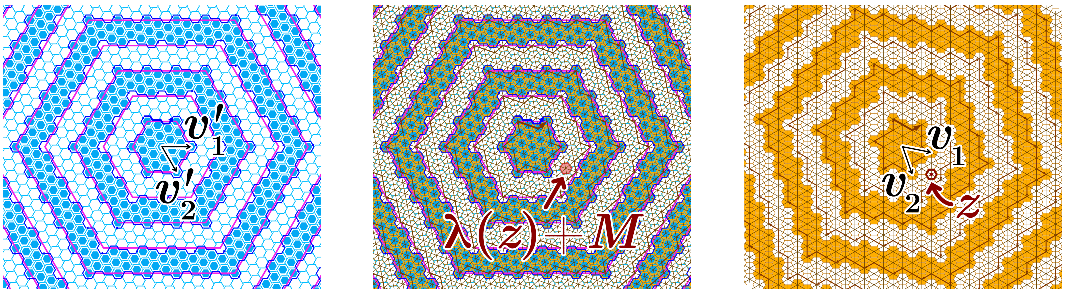


Figure 8 The linear map λ between the turedo world and the oritatami world induces a tilt between the concentric balls in the both worlds. This simulation tilt must be compensated by providing to the simulated turedo a pair of vectors as an input, so that it fills a proper discretization of the oritatami world balls when simulated: (left) the shortest radius vectors v'_1, v'_2 of a ball in the oritatami world that can be mapped exactly in the turedo world – (right) when the two corresponding vectors v_1, v_2 in the turedo world are supplied to the turedo as an input, the turedo can use them to build a proper discretization of large enough balls in the oritatami world – (middle) both turedo and oritatami worlds superposed (the target balls are drawn in purple and the discretized turedo ones in blue).

5 Characterization of Possible Densities of Limit Configurations

We can define the (upper) density $\overline{d}(c)$ of non-blank cells in configuration c as follows:

$$\overline{d}(c) = \limsup_n \frac{\#\{z \in B(n) : c(z) \neq \bot\}}{b(n)}.$$

This choice is natural and gives a translation-invariant notion, but it is not unique (we could replace the sequence $(B(n))_n$ by another Følner sequence [7]). The problem is that, in a simulation, the lattice of cells is distorted into a macro-lattice of macro-cells in such a way that the macro-balls do not have the same shape as genuine balls, as shown in Fig. 8. Said differently, the reference Følner sequence is distorted into another one and this can change the density. To circumvent this problem and produce more robust results, we will consider all possible linearly distorted balls from the start: for any pair $v_1, v_2 \in \mathbb{Z}^2$ of non-colinear vectors, we consider the (upper) density $\overline{d}_{v_1,v_2}$ of non-blank state after distortion of the lattice by the pair v_1 and v_2.

We first prove that the computational complexity of $\overline{d}_{v_1,v_2}(c)$ is Π_2-bounded as soon as c is produced as the limit of a computable process on finite configurations such that the set of non-blank positions is monotonically increasing and with diameter growing in a computable way. This bound applies to turedos but also all systems cited in section 1.

▶ **Lemma 5.1** (Densities of any self-assembling systems are Π_2). *Let c^∞ be the limit configuration reached from some finite seed by some system among oritatami, turedos, freezing cellular automata or directed aTAM. Then for any pair of non-colinear vectors v_1, v_2, the upper density $\overline{d}_{v_1,v_2}(c^\infty)$ is a Π_2-computable number.*

For non-deterministic systems (both turedos and aTAM), we can state a similar lemma saying that, starting from any finite seed, there is always one orbit converging to a configuration with Π_2 density.

Arbitrarily dense simulation. The next theorem is a stronger version of Theorem 1.1, enforcing a *constant* and arbitrarily large density inside each macrocell of the oritatami simulation of a given turedo. Precisely, if we consider the *cell partition* of the oritatami

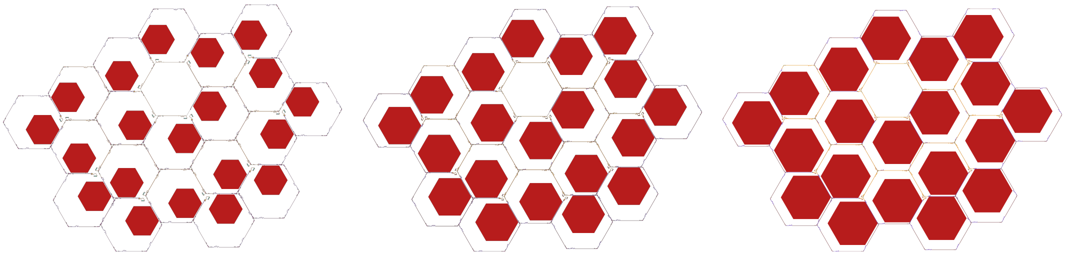


Figure 9 Increasing the density by folding a filled hexagon inside the macrocell expanded by 50, 100 and 200 extra $2n$-periods on each side. Actual oritatami simulation. Zoom in for details

plane into disjoint identical copies of a *macrocell tile* M induced by the map λ from the turedo world to the oritatami world, where each copy $\lambda(z) + M$ covers exactly the macrocell corresponding to the turedo position z (see Fig. 8), then:

▶ **Theorem 5.2.** *For any turedo $\mathcal{T}$ of radius 1, and for any $\epsilon > 0$, there exists an oritatami system of delay 3 that simulates $\mathcal{T}$ and such that the number of occupied positions in each macrocell tile $\lambda(z) + M$ in the oritatami limit configuration is exactly k for all non-$\perp$ position z of the turedo limit configuration (and 0 for $\perp$ position), with $k \geqslant (1 - \epsilon) \cdot \#M$.*

This result is obtained by 1) expanding of the macrocell with a straight line of length L in the middle of each side so that the empty triangles between the macrocells become negligible and 2) inserting a sequence in the scaffold that folds into a filled hexagon of radius $L(1 + \alpha)$ inside the space freed inside the macrocell by the expansion. The factor $\alpha > 0$ is necessary to account for the increase of the exit pocket induced by the increase of the side length (more transcript needs to fit into the pocket) (see Fig. 9). Picking L large enough concludes the proof. The case of density 1 is treated separately by an ad hoc solution.

Arbitrary Π_2-density. We conclude with the construction of a turedo of radius 1 that is able to produce limit configurations with any possible density when starting from the appropriate finite configuration. By possible density we mean any real number $d \in [0, 1]$ which is Π_2-computable [25], *i.e.* such that there exists a computable sequence of rational numbers (q_n) with $d = \limsup_n q_n$. The construction is rather technical but the overall idea is simple (see Fig. 7b): at step n, leave a large annulus empty then densely fill another large annulus in such a way that the surface ratio between these annuli is q_n and that their sizes are large enough to dominate all the previously constructed annulus in anterior steps. The exact sequence of annuli is computed by the turedo in a sublinearly growing (hence negligible) corridor.

▶ **Theorem 5.3.** *There exists a turedo of radius 1 such that for any Π_2-computable number $d \in [0, 1]$ and any pair of non-colinear vectors v_1, v_2, there is a finite initial global state such that the limit tape configuration c^∞ reached from it verifies: $\overline{d}_{v_1,v_2}(c^\infty) = d$.*

The Π_2-computability limitation is unavoidable as shown in Lemma 5.1, hence our result is optimal and actually gives a characterization of densities of limit configurations of continuous sequential self-avoiding systems (resp. turedo, resp. oritatami) started from finite configurations. Using our simulation framework and Theorem 5.2 we directly deduce Theorem 1.3.

References

1 Sebastián Barbieri, Jarkko Kari, and Ville Salo. The group of reversible Turing machines. In *Cellular Automata and Discrete Complex Systems*, pages 49–62. Springer International Publishing, 2016. doi:10.1007/978-3-319-39300-1_5.

2 Florent Becker, Diego Maldonado, Nicolas Ollinger, and Guillaume Theyssier. Universality in freezing cellular automata. In *Sailing Routes in the World of Computation - 14th Conference on Computability in Europe, CiE 2018, Kiel, Germany, July 30 - August 3, 2018, Proceedings*, pages 50–59, 2018. doi:10.1007/978-3-319-94418-0_5.

3 Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking ii: Classifications of cellular automata. *Theor. Comput. Sci.*, 412(30):3881–3905, 2011. doi:10.1016/j.tcs.2011.02.024.

4 Erik D. Demaine, Jacob Hendricks, Meagan Olsen, Matthew J. Patitz, Trent A. Rogers, Nicolas Schabanel, Shinnosuke Seki, and Hadley Thomas. Know when to fold 'em: Self-assembly of shapes by folding in oritatami. In *DNA Computing and Molecular Programming - 24th International Conference, DNA 24, Jinan, China, October 8-12, 2018, Proceedings*, volume LNCS 11145, pages 19–36, 2018. doi:10.1007/978-3-030-00030-1_2.

5 D. Doty, J. H. Lutz, M. J. Patitz, R. T. Schweller, and S. M. Summer. The tile assembly model is intrinsically universal. In *Proceedings of the 53rd Annual Foundations of Computer Science (FOCS)*, 2012.

6 David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. The tile assembly model is intrinsically universal. In *FOCS2012: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 302–310, 2012.

7 Erling Følner. On groups with full banach mean value. *MATHEMATICA SCANDINAVICA*, 3:243, December 1955. doi:10.7146/math.scand.a-10442.

8 Cody Geary, Guido Grossi, Ewan K. S. McRae, Paul W. K. Rothemund, and Ebbe S. Andersen. RNA origami design tools enable cotranscriptional folding of kilobase-sized nanoscaffolds. *Nature Chemistry*, 13:549–558, 2021.

9 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In *MFCS2016: Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science*, volume 58 of *LIPIcs*, pages 43:1–43:14, 2016.

10 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Oritatami: A computational model for molecular co-transcriptional folding. *International Jounal of Molecular Sciences*, 9(2259), 2019. doi:10.3390/ijms20092259.

11 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinonsuke Seki. Proving the Turing universality of oritatami cotranscriptional folding. In *ISAAC 2018: Proceedings of the 29th International Symposium on Algorithms and Computation*, volume 123 of *LIPIcs*, pages 23:1–23:13, 2018.

12 Cody Geary, Paul W. K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345:799–804, 2014.

13 E. Goles, N. Ollinger, and G. Theyssier. Introducing freezing cellular automata. In J. Kari, I. Törmä, and M. Szabados, editors, *Exploratory Papers of Cellular Automata and Discrete Complex Systems (AUTOMATA 2015)*, pages 65–73, 2015.

14 Petr Kůrka. *Topological and symbolic dynamics*. Société Mathématique de France, 2003.

15 James I. Lathrop, Jack H. Lutz, Matthew J. Patitz, and Scott M. Summers. Computability and complexity in self-assembly. *Theory Comput. Syst.*, 48(3):617–647, 2011. doi:10.1007/s00224-010-9252-0.

16 Diego Maldonado, Anahí Gajardo, Benjamin Hellouin de Menibus, and Andrés Moreira. Nontrivial turmites are Turing-universal. *J. Cell. Autom.*, 13(5-6):373–392, 2018. URL: http://www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-13-number-5-6-2018/jca-13-5-6-p-373-392/.

17 Yusei Masuda, Shinnosuke Seki, and Yuki Ubukata. Towards the algorithmic molecular self-assembly of fractals by cotranscriptional folding. In *CIAA2018: the 23rd International Conference on Implementation and Application of Automata*, volume 10977 of *LNCS*, pages 261–273. Springer, 2018.

18 Nicolas Ollinger and Guillaume Theyssier. Freezing, bounded-change and convergent cellular automata. *CoRR*, abs/1908.06751, 2019. `arXiv:1908.06751`.

19 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.

20 Daria Pchelina, Nicolas Schabanel, Shinnosuke Seki, and Yuki Ubukata. Simple intrinsic simulation of cellular automata in oritatami molecular folding model. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *LATIN 2020: Theoretical Informatics - 14th Latin American Symposium, São Paulo, Brazil, January 5-8, 2021, Proceedings*, volume 12118 of *Lecture Notes in Computer Science*, pages 425–436. Springer, 2020. `doi:10.1007/978-3-030-61792-9_34`.

21 Paul W. K. Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2:2041–2053, 2004.

22 Nicolas Schabanel. Simple OS simulator, 2021. URL: `http://perso.ens-lyon.fr/nicolas.schabanel/OSsimulator/`.

23 Nicolas Schabanel and Shinnosuke Seki. Turedo to oritatami compiler, 2022. URL: `https://hub.darcs.net/turedo2oritatami/turedo2oritatami`.

24 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, 1998.

25 Xizhong Zheng and Klaus Weihrauch. The arithmetical hierarchy of real numbers. In Mirosław Kutyłowski, Leszek Pacholski, and Tomasz Wierzbicki, editors, *Mathematical Foundations of Computer Science 1999*, pages 23–33, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

A Transcript, Folding meter and Pocket

In our design, the oritatami transcript is periodic, and one period folds into one macrocell. The period is divided semantically in five parts:

Transcript = Scaffold · Read · Write · Speedbump · Exit

Scaffold hardcodes the "skeleton" of the macrocell and folds clockwise. Read folds around the scaffold counterclockwise. It reads the states of the adjacent macrocells, which induces a shift of the transcript equal to the total weight of the bits read with value `1`. Write folds on top of the Read layer clockwise, and, according to the shift read, writes the bits to be output on each side and marks the exit side. Speedbump annihilates the shift using a process similar to [20]. Finally, Exit folds on top of the Write layer counterclockwise until it reaches the "exit mark" that has been placed by the Write layer.

Each of the Read, Write, Exit layers have the same periodic structure that we call a folding meter. A *n-folding meter* is a $4n$-periodic transcript with a period R of the form:

$$\begin{aligned}\mathsf{R} = {} & \mathsf{Rt}_0,\ \mathsf{Rt}_1,\ \mathsf{Rt}_2,\ \mathsf{Rp}_3,\ \dots,\ \mathsf{Rp}_{n-1},\ \mathsf{Rb}_n,\ \mathsf{Rb}_{n+1},\ \mathsf{Rb}_{n+2},\ \mathsf{Rq}_{n+3},\ \dots,\ \mathsf{Rq}_{2n-1},\\ & \mathsf{Rt}_{2n},\ \mathsf{Rt}_{2n+1},\ \mathsf{Rt}_{2n+2},\ \mathsf{Rp}_{2n+3},\ \dots,\ \mathsf{Rp}_{3n-1},\ \mathsf{Rb}_{3n},\ \mathsf{Rb}_{3n+1},\ \mathsf{Rb}_{3n+2},\ \mathsf{Rq}_{3n+3},\ \dots,\ \mathsf{Rq}_{4n-1}\end{aligned}$$

where the letters `t` and `b` stand for *top* and *bottom*. The internal interactions are:

$$\underbrace{\mathsf{R}_i \heartsuit \mathsf{R}_{-i-1},\quad \mathsf{R}_i \heartsuit \mathsf{R}_{-i-2},}_{\text{Going down}}\quad \underbrace{\mathsf{R}_{n+i} \heartsuit \mathsf{R}_{n-i},\quad \text{and } \mathsf{R}_{n+i} \heartsuit \mathsf{R}_{n-i-1}}_{\text{Going up}}\quad \text{for all } i. \tag{1}$$

They ensure that the folding meter will either:

- *follow a border* if all of its beads bind to every bead on the border. Two examples are the Read layer along the right border in Fig. 5, and the Write layer along the left border in Fig. 11. This process allows as well to stack several n-folding meters on top of each other in opposite direction and that are in-sync, i.e. such that one's p-parts face other's q-parts. As an example, observe the folding of the three layers Read, Write and Exit at the top right of the write pocket in Fig. 11;

- or *fold upon itself in the manner of the "folding meter" tool*, when reaching the bottom of a *pocket* that no longer attract the layer. Two examples are the Read layer in Fig. 5b or the Write layer in Fig. 11. Indeed, in Fig. 11, the beads rp15..12 do not attract the beads Wb26..28 which thus fold upwards thanks to the interactions listed in Eq. (1) yielding to a switchback pattern that continues until the Write layer reaches the right side, to which the Write layer is attracted and thus resumes following the border.

As the binding between the switchbacks of a n-folding meter are strong, they can flatten any custom interactions encoded internally in either the p- or the q-beads of an n-folding meter as long as these interactions are weak, i.e. do not involve more than 3 bonds per bead. This allows us to hide or expose on-demand specific behaviors when the binding with the lower layer is weak enough: for instance, in Fig. 10, the binding between the p-parts of the Write and Read layers is weak enough to let the p-part of the Write layer fold into various two-spikes patterns encoding 0 or 1 that flatten anywhere else in the macrocell.

In this article, $n = 26$. Note that each folding meter is essentially $2n$-periodic with period $(\mathsf{t}, \mathsf{p}, \mathsf{b}, \mathsf{q})$. This $2n$-period is repeated twice only to prevent unwanted interactions when in switchback form. This is why everywhere in the paper the true unit of length is $2n$, half-period of the folding meter, and not a full period. Furthermore every bead type Ri in a folding meter R behaves the same as the beadtype R$(i + 2n)$ = R$(i + 52)$. For this reason, we will adopt the following notation: given a folding meter R, R$\overline{i}$ will refer to either bead types Ri or R$(i + 2n)$; for instance R$\overline{12}$ refers to both bead types R12 and R64.

Notations. For any pair of integers $x \geqslant 0$ and $y \geqslant 1$:

- x.nextMultiple(of: y) = $y\lceil x/y\rceil$ is the least multiple of y larger or equal to x
- x.complement(to: y) = $y\lceil x/y\rceil - x$ so that $x + x$.complement(to: y) = x.nextMultiple(of: y)

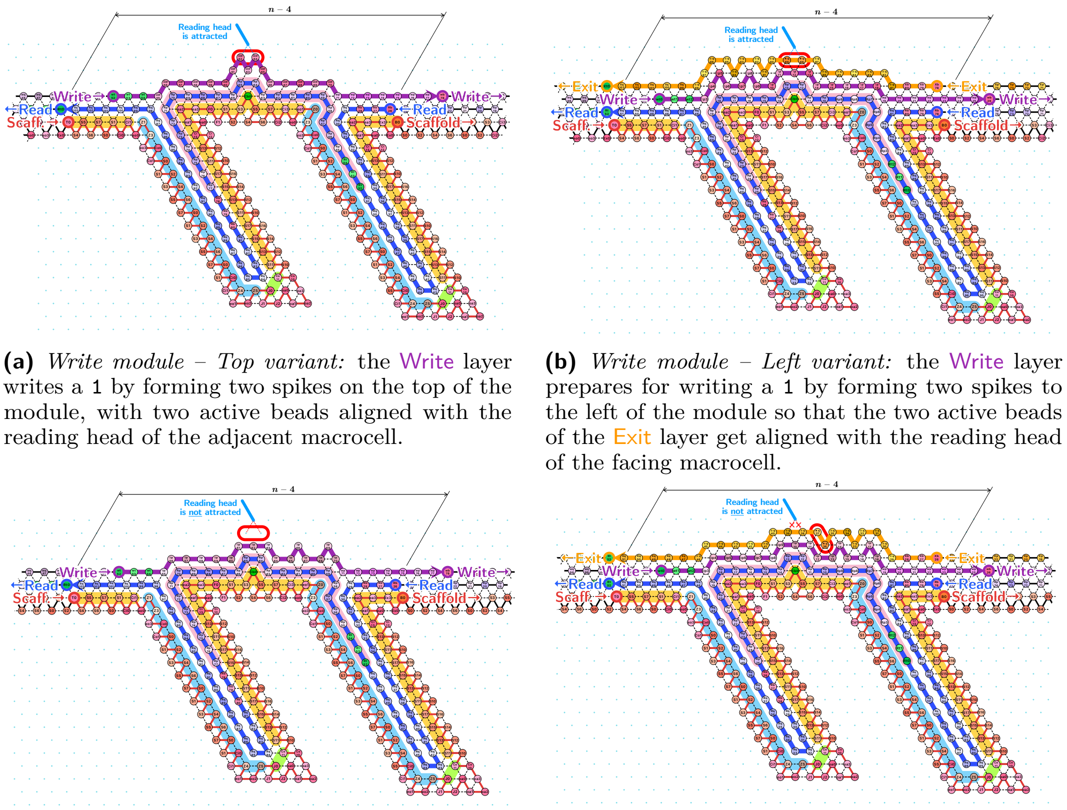
B The write block

As seen in Fig. 6, the *write block* on each side of a macrocell consists in an alternation of $q + 1$ write pockets of capacity $2n\mathcal{W}$ beads, and q write modules, one for each of the q bits to be written. Each write pocket hides the $\mathcal{W}$ entries of the $(\mathcal{W} + 1)$-long transition tables that are unused, while the write module writes the selected entry. Let us start by describing the write module.

B.1 Write module

Write modules are the places where the oritatami writes the bits output on each side. Every side of a macrocell is provided with q write modules, each of which is responsible for one of the q bits to be output. Precisely, this module places two beads of special type (circled in red in figures) at a designated readable site to write a 1 (Figs. 10a and 10b), or deliberately out of the site so that they cannot attract the reading head no matter what types they are to write a 0 (Figs. 10c and 10d).

Depending on whether the module is located before of after the side by which the oritatami will exit the macrocell, the module will be covered or not by the Exit layer, leading to the 4 possible configurations in total presented in Fig. 10. As the side by which the oritatami system will exit is known as soon as the states of the neighboring macrocell are read, we can



(a) *Write module – Top variant:* the Write layer writes a 1 by forming two spikes on the top of the module, with two active beads aligned with the reading head of the adjacent macrocell.

(b) *Write module – Left variant:* the Write layer prepares for writing a 1 by forming two spikes to the left of the module so that the two active beads of the Exit layer get aligned with the reading head of the facing macrocell.

(c) *Write module – Right variant 1:* on a side located before the exit that will be taken later, the Write layer writes a 0 by forming two spikes to the right of the module; as the Exit layer will exit before reaching this position, the reading positions will stay empty, which will be interpreted as a 0 by the reading head of the facing macrocell.

(d) *Write module – Right variant 2:* the Write layer prepares for writing a 0 by forming two spikes to the right of the module, so that the two active beads of the Exit layer get misaligned with the reading head of the facing macrocell.

Figure 10 The four variants of the Write module: **(a,b)** writing a 1 and **(c,d)** writing a 0.

use, in the Write layer description, the appropriate variant of the encoding for each bit on each bit according of the relative position of the module with the exit direction to be taken:

- In the case where the Write layer is to be covered, the Write layer folds into two spikes either to the left (Fig. 10b) or the right (Fig. 10d) of the blue hill at the center of the module; then, when the Exit layer folds from right to left, the two special bead types ★$\overline{32}$ and ★$\overline{33}$ in the Exit layer will either be placed at the top of the hill (where they will attract the reading head) or hidden in the right side of the hill (where they cannot attract the reading head), which will be interpreted by the facing macrocell as a 1 or a 0 respectively.
- Otherwise, the bit 0/1 is encoded directly either by two big spikes bearing the special attracting bead types |⊔|$\overline{10}$ and |⊔|$\overline{12}$ at the top of the hill, which will be read as a 1 (Fig. 10a), or by two spikes to the left of the hill leaving the designated site empty, which will be read as a 0 (Fig. 10c).

In order for the Write layer to adopt these peculiar configurations, we need it to take its independence from the underlying Read layer. This is accomplished thanks to the two pitfalls surrounding the write module. Each of them have a capacity of n beads exactly, which induces a phase difference of n between the Read and the Write layers, resulting in the p-part of the Write layer to fold on top of the p-part of Read layer over the write modules. These two p-parts have specific interactions between them allowing the specific configurations to be folded there and only there.

B.2 Write pocket

Write pocket operation. Its primary purpose is to hide the $\mathcal{W}$ unused entries of the $(\mathcal{W}+1)$-long transition tables as can be observed in Fig. 6. These pockets are placed between the write modules so that only the entries to be written are exposed on the border, at the locations of the write modules; all the others are hidden in the write pockets. The Read layer does not fill the write pocket but simply "coats" its border. The Write layer, however, enters it and folds inside into its compact switchback form, hiding away the $\mathcal{W}$ unused entries of each transition table, encoded each in one p-part of the Write layer. Finally, the Exit layer simply jumps over it by folding into a hardcoded bridge to get across its entrance.

As for the read pocket, write pockets are also used in the interchange blocks to switch between the three variants of the write layer Write1 ↔ Write12 ↔ Write2 (see Fig. 4).

Write pocket design. This pocket differs from the read pocket in three ways: 1) as opposed to the read pocket, both layers Read and Write will enter the pocket unconditionally; 2) the Read and Write layers will enter the pocket from opposite directions; 3) only the Write layer fills the pocket: the Read layer must not fill the pocket. It follows that:

- Been "coated" by the Read layer, both sides of the write pocket entrance will attract the Write layer. To avoid unwanted interactions, we need thus to make the entrance wider. It follows that the Exit layer will not be able to jump over the entrance as the Write layer does over the read pocket. Fortunately, as the Exit layer is never shifted, we can hardcode a bridge $\mathtt{G}\overline{4..16}$ in this layer at this precise location to solve this issue (see top of Fig. 11).
- Furthermore, as the Read layer will enter the pocket with an arbitrary shift, the interactions of the pocket with the Write layer cannot be directly hardcoded. As illustrated in Fig. 5, a pocket has essentially three kinds of interactions with the layer that fills it: 1) full attraction (highlighted in yellow), 2) no attraction (in blue), and 3) localised specific attractions (in green). We create similar interactions using two mechanisms: A) two pitfalls are located at the bottom left and at the middle right of the pocket (see Fig. 11) that introduce and then cancel a phase difference between the Read and Write layers; B) the bottom and bottom right borders of the pocket are moved out of reach of the Write layer switchbacks. A) and B) ensure at the bottom right that the Write layer is not attracted by the bottom nor opposite border while it folds in switchbacks. Specifically programmed interactions between the in-sync write bead types $\mathtt{@b}\overline{78..80}\mathtt{@q}\overline{81..83}$ and the out-of-sync read bead types $\mathtt{rp}\overline{24..25}\mathtt{rb}\overline{26..28}\mathtt{rq}\overline{29}$ that appear just before the second pitfall in the middle of the right border, ensure that the Write layer glues back to the border once it has ended its switchback pattern and not earlier.

Fitting all these constraints together contributes to the choice of $n = 26$ for the period of our folding meters.

Geometry. The pocket geometry is determined by four integer parameters: its width w, height k, remainder $\rho < 2k + 1$, and extension ℓ. The two "bubbles" to the right of the pocket are used to keep synchronised the three layers Read, Write and Exit. Their sizes are determined by two extra integer parameters x and y which are adjusted as follows:

(a) The length of the Write layer path from Wt$\overline{0}$/Wb$\overline{26}$ to Wt$\overline{0}$ inside the upper bubble should be equal to 0 modulo $2n$ if ρ is even or equal to n modulo $2n$ if ρ is odd. The total length is $2(x + w + 17) + \rho n$; thus, x should be set to: $\mathbf{x} = (\mathbf{w} + \mathbf{17})$.complement(to: n).

(b) The length of the Read layer path from Rt$\overline{0}$/Rb$\overline{26}$ to Rb$\overline{26}$ inside the lower bubble should be equal to 0 modulo $2n$ if ρ is odd, or equal to n modulo $2n$ if ρ is even. The total length is $2(w + y + 12) + (2k + 1 - \rho)n$; thus, we set: $\mathbf{y} = (\mathbf{w} + \mathbf{12})$.complement(to: n).

(c) Lastly, in order to avoid collision with other modules, we set the extension ℓ so that the pocket module ends to the right of the two bubbles, that is: $\ell = (\mathbf{2w} + \max(\mathbf{x} + \mathbf{8}, \mathbf{y} - \mathbf{3}))$.nextMultiple(of: $2n)/\mathbf{2n}$.

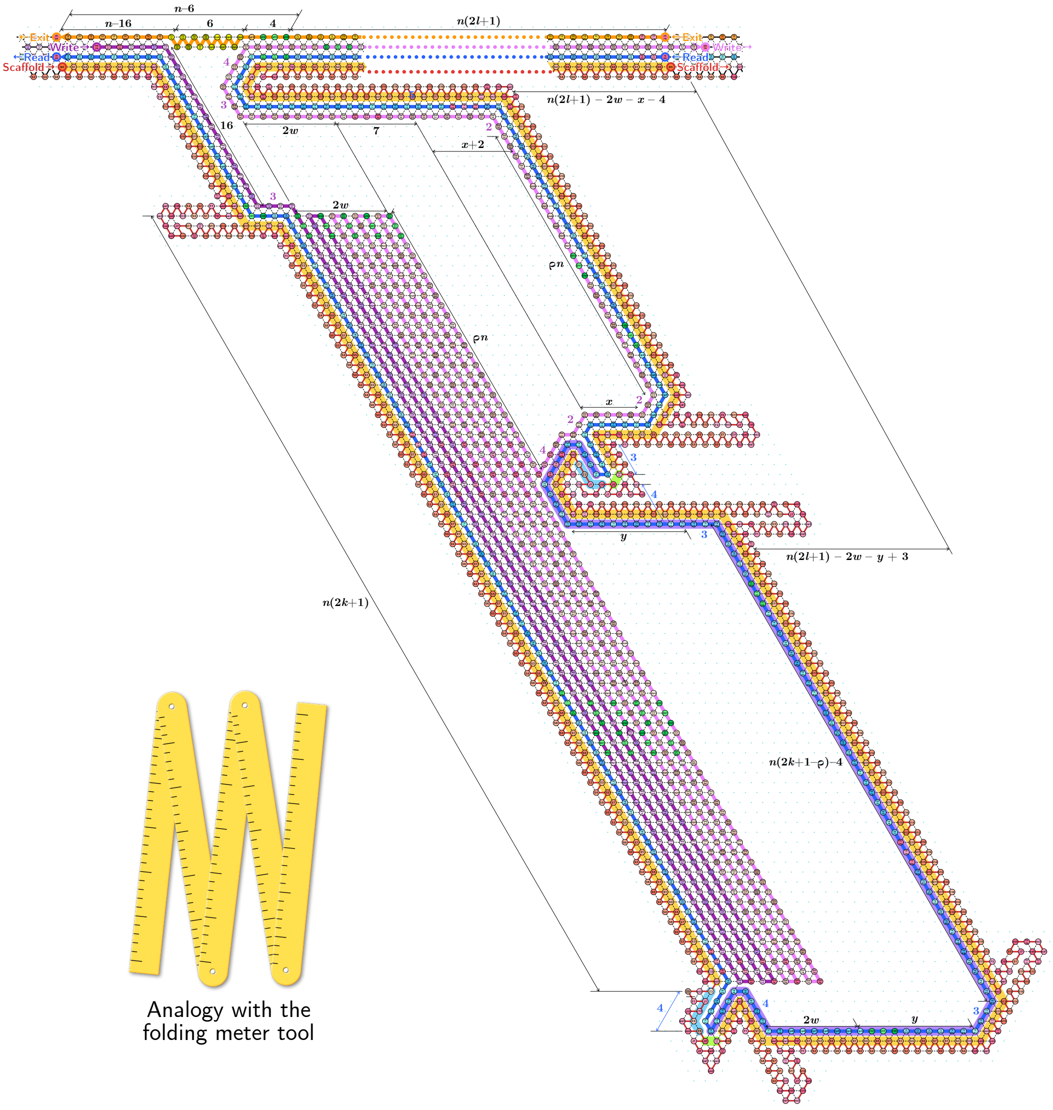


Figure 11 Write pocket. Given w, parameters x, y and ℓ must be adjusted so as to match the period of the folding meter. The part of the Read layer which is desynchronized with the Write layer is highlighted in purple.

Capacity. The *capacity* of a write pocket is defined as the length of the path taken by the Write layer from the leftmost $\mathsf{Wt}\overline{0}$ to the rightmost $\mathsf{Wt}\overline{0}$, and it is determined by the three independent parameters k, w, ρ with $\rho < 2k+1$, and one dependent parameter ℓ as:

$$\mathsf{capacity}(w,k,\rho,\ell) = 2n((2k+1)w+\rho) + 2(w+17).\mathsf{nextMultiple}(\mathsf{of}\colon\ n) + 2n\ell.$$

Building a write pocket with a given capacity. Various parameters can yield the same capacity, thus we aim for the parameters that yield the shortest transcript. The length of the transcript is given by the Read layer, whose asymptotic length is $\sim 4nk + 6w$. Minimizing this value subject to a fixed asymptotic capacity of $2w(2k+1)n$ yields to the ideal ratio of $w \sim 2nk/3$. Now, to obtain a write pocket of target capacity $2n\mathcal{W}$ we proceed as follows:

- solving $\mathsf{capacity}(w = 2nk/3, k, \rho = 0, \ell = 0) = 2n\mathcal{W}$ yields a suggested value for k of:

$$k := \max\left(0, \left\lfloor \frac{\sqrt{12n\mathcal{W} + n^2 + 4n - 224}}{4\,n} - \frac{1}{2n} - \frac{1}{4} \right\rfloor\right)$$

- we then set w by solving $2n\mathcal{W} = \mathsf{capacity}(w, k, \rho = 0, \ell \sim w/n)$ which yields:

$$w := \max\left(1, \left\lfloor \frac{n\mathcal{W} - 19}{n(2k+1)+2} \right\rfloor\right)$$

- x, y, and ℓ are then computed according to the formulas (a), (b), and (c).
- we conclude by setting:

$$\rho := \max\left(0, 2n\mathcal{W} - \overbrace{\Big(2n((2k+1)w+1) + 2(w+17).\mathsf{nextMultiple}(\mathsf{of}\colon\ n) + 2n\ell\Big)}^{\mathsf{capacity}(w,k,\rho=0,\ell)}\right) \Big/ n,$$

  if $\rho > 2k$, then rerun the two last steps with $w := w + 1$.

This ensures that: $2n\mathcal{W} \leqslant \mathsf{capacity}(w,k,\rho,\ell) \leqslant 2n(\mathcal{W}+2)$ and that k, w, and ℓ are $O(\sqrt{\mathcal{W}})$.

Compact Representation for Matrices of Bounded Twin-Width

Michał Pilipczuk
Institute of Informatics, University of Warsaw, Poland

Marek Sokołowski
Institute of Informatics, University of Warsaw, Poland

Anna Zych-Pawlewicz
Institute of Informatics, University of Warsaw, Poland

Abstract

For every fixed $d \in \mathbb{N}$, we design a data structure that represents a binary $n \times n$ matrix that is d-twin-ordered. The data structure occupies $\mathcal{O}_d(n)$ bits, which is the least one could hope for, and can be queried for entries of the matrix in time $\mathcal{O}_d(\log \log n)$ per query.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases twin-width, compact representation, adjacency oracle

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.52

Related Version *Full Version*: https://arxiv.org/abs/2110.08106 [14]

Funding This work is a part of projects that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreements No. 948057 (Mi. Pilipczuk, M. Sokołowski) and 714704 (A. Zych-Pawlewicz).

1 Introduction

Consider a *binary* matrix M, that is, one with entries in $\{0,1\}$. Given two consecutive columns c_1, c_2 of M, the operation of *contracting* those columns consists of replacing them with a single column c with entries reconciled as follows. If in a row r columns c_1, c_2 agree, that is, both contain symbol 0 or both contain symbol 1, then in column c in row r we put the same symbol. Otherwise, we put a special mismatch symbol $\perp$. Contracting consecutive rows is defined analogously. Contractions of rows and columns can be then applied further with the rule that reconciling any entry with the mismatch symbol $\perp$ again results in $\perp$. For a nonnegative integer $d \in \mathbb{N}$, we say that M is *d-twin-ordered* if M can be contracted to a 1×1 matrix in such a way that at every point during the process, every row and every column contains at most d symbols $\perp$. Finally, the *twin-width* of M is the least d for which one can permute rows and columns of M so that the resulting matrix is d-twin-ordered.

The notion of twin-width was introduced very recently by Bonnet et al. in [5], and has immediately gathered immense interest. As shown in [5] and in multiple subsequent works [1, 2, 3, 4, 6, 9, 10], twin-width is a versatile measure of complexity not only for matrices, but also for permutations and for graphs by considering a suitable matrix representation, which in the latter case is just the adjacency matrix. In particular, for every fixed $t \in \mathbb{N}$, graphs excluding K_t as a minor and graphs having cliquewidth at most t have bounded twin-width, which means that the concept of boundedness of twin-width is a vast generalization of boundedness of cliquewidth that does not assume tree-likeness of the structure of the graph. As shown in the aforementioned works, this generalization is combinatorially rich [1, 5, 9], algorithmically useful [2, 4, 5], and exposes deep connections with notions studied in finite

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 52; pp. 52:1–52:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

model theory [3, 5, 6, 10]. In particular, assuming a suitable contraction sequence is provided on input, model-checking First-Order logic on graphs of bounded twin-width can be done in linear fixed-parameter time [5].

One of the fundamental directions in the work on twin-width is that of estimating the asymptotic *growth* of considered classes of objects. It has been proved in [1] that the number of distinct graphs on vertex set $\{1,\ldots,n\}$ that have twin-width at most d is bounded by $2^{\mathcal{O}_d(n)} \cdot n!$, which renders the class of graphs of twin-width at most d *small.* Similarly, the number of distinct $n \times n$ binary matrices that are d-twin-ordered is upper bounded by $2^{\mathcal{O}_d(n)}$ [3] (see also the proof of Lemma 16).

The latter result raises a natural data structure question, which we address in this work. In principle, a binary $n \times n$ matrix of twin-width at most d can be encoded using $\mathcal{O}_d(n)$ bits, just because the number of such matrices is bounded by $2^{\mathcal{O}_d(n)}$. However, would it be possible to design such a representation so that it is algorithmically useful in the following sense: the representation may serve as a data structure that supports efficient queries for entries of the matrix. This question originates from the area of *compact representations*, see for instance the work on such representations for graphs of bounded cliquewidth [11].

Let us review some solutions to the above problem that to smaller or larger extent follow from existing literature. The quality of a representation is measured by the number of bits it occupies and the worst-case time complexity of a query for an entry. Here, we assume the standard word RAM model with word length $\mathcal{O}(\log n)$.

- Storing the matrix explicitly is a representation with bitsize $\mathcal{O}(n^2)$ and query time $\mathcal{O}(1)$.
- In [1], Bonnet et al. presented an *adjacency labelling scheme* for graphs of bounded twin-width, which can be readily translated to the matrix setting. This scheme assigns to each row and each column of the matrix a *label* – a bitstring of length $\mathcal{O}_d(\log n)$ – so that the entry in the intersection of a row and a column can be uniquely decoded from the pair of their labels. In [1] the time complexity of this decoding is not analyzed, but a straightforward implementation runs in time linear in the length of labels. This gives a representation with bitsize $\mathcal{O}_d(n \log n)$ and query time $\mathcal{O}_d(\log n)$.
- It follows from the results of [2] that if matrix M is d-twin-ordered, then the entries 1 in M can be partitioned into $\ell = \mathcal{O}_d(n)$ rectangles, say $R_1,\ldots,R_\ell$ (see Lemma 10 for a proof). This reduces our question to *2D orthogonal point location*: designing a data structure that for a given point in $(i,j) \in \{1,\ldots,n\}^2$, may answer whether (i,j) belongs to any of the rectangles $R_1,\ldots,R_\ell$. For this problem, Chan [7] designed a data structure with bitsize $\mathcal{O}(n \log n)$ and query time $\mathcal{O}(\log\log n)$ assuming $\ell = \mathcal{O}(n)$. So we get a representation of M with bitsize $\mathcal{O}_d(n \log n)$ and query time $\mathcal{O}_d(\log\log n)$.
- For 2D orthogonal point location one can also design a simple data structure by persistently recording a sweep of the square $\{1,\ldots,n\}^2$ using a B-ary tree for $B = n^\varepsilon$, for any fixed $\varepsilon > 0$. This gives a representation with bitsize $\mathcal{O}_d(n^{1+\varepsilon})$ and query time $\mathcal{O}(1/\varepsilon)$. See Appendix A for details.

Note that in all solutions above, the bitsize of the representation is $\Omega(n \log n)$, and thus does not reach the information-theoretic limit of $\Theta_d(n)$.

Our result. We design a compact representation for d-twin-ordered matrices that simultaneously occupies $\mathcal{O}_d(n)$ bits and offers query time $\mathcal{O}_d(\log\log n)$. The result is summarized in the statement below.

▶ **Theorem 1.** *Let $d \in \mathbb{N}$ be a fixed constant. Then for a given binary $n \times n$ matrix M that is d-twin-ordered one can construct a data structure that occupies $\mathcal{O}_d(n)$ bits and can be queried for entries of M in worst-case time $\mathcal{O}(\log\log n)$ per query. The construction time is $\mathcal{O}_d(n \log n \log\log n)$ in the word RAM model, assuming M is given by specifying $\ell = \mathcal{O}_d(n)$ rectangles $R_1,\ldots,R_\ell$ that form a partition of symbols 1 in M.*

The proof of Theorem 1 proceeds roughly as follows. Consider a parameter m that divides n and a partition of the given matrix M into $(n/m)^2$ *zones* – square submatrices – each of which is induced by m consecutive rows and m consecutive columns. Such a partition is called the *regular* (n/m)*-division.* Even though the total number of zones in the regular (n/m)-division is $(n/m)^2$, one can use the connections between the notions of being twin-ordered and that of *mixed minors*, developed in [5], to show that actually there will be only $\mathcal{O}_d(n/m)$ *different* zones, in the sense that zones are considered equal if they have exactly the same values in corresponding entries.

Our data structure describes the zones in the regular (n/m)-divisions of M for m ranging over a sequence of parameters $m_0 > m_1 > \ldots > m_\ell$ for $\ell = \mathcal{O}(\log\log n)$, where m_j divides m_i whenever $i \le j$. Roughly speaking, we set $m_0 = n$ and $m_i = m_{i-1}^{2/3}$ for $i \ge 1$, though for technical reasons we resort to the recursion $m_i = m_{i-1}/2$ once m_i reaches the magnitude of $\log^3 n$. Each different zone present in the regular (n/m_i)-division is represented by a square matrix consisting of $(m_i/m_{i+1})^2$ pointers to representations of its subzones in the regular (n/m_{i+1})-division. When we reach $m_i < c_d \cdot \log n$ for some small constant c_d depending on d, we stop the construction and set $\ell = i$. At this point the number of different zones present in the regular (n/m_ℓ)-division of M is strongly sublinear in n, because we have such an upper bound on the total number of different $(c_d \log n) \times (c_d \log n)$ binary matrices that are d-twin-ordered, and $n/m_\ell \le c_d \log n$. Therefore, all those matrices can be stored in the representation explicitly within bitsize $\mathcal{O}_d(n)$.

The query algorithm is very simple: just follow appropriate pointers through the $\mathcal{O}(\log\log n)$ levels of the data structure and read the relevant entry in a matrix stored explicitly in the last level. The analysis of bitsize is somewhat more complicated, but crucially relies on the fact that in the ith level, it suffices to represent only $\mathcal{O}_d(n/m_i)$ different matrices that are zones in the (n/m_i)-division.

We remark that the idea of dividing the given matrix into a number of polynomially smaller zones, and describing them recursively, is also the cornerstone of the approach used by Chan for the orthogonal point location problem in [7]. However, when it comes to details, his construction is quite different and technically more complicated. For instance, in [7] the recursion can be applied not only on single zones, but also on wide or tall strips consisting of several zones, or even submatrices induced by non-contiguous subsets of rows and columns. The conceptual simplification achieved here comes from the strong properties implied by the assumption that the matrix is d-twin-ordered, which is stronger than the assumption used by Chan that the symbols 1 in the matrix can be partitioned into $\mathcal{O}(n)$ rectangles.

Organization of the paper. In Section 2 we define the twin-width of matrices formally and recall a number of notions related to d-twin-ordered matrices. In Section 3, we prove several new structural properties of those matrices. These properties are exploited in Section 4 to derive an efficient and compact representation of d-twin-ordered matrices, completing the non-constructive part of Theorem 1. The efficient algorithm for construction of the data structure is deferred to the full version of the paper [14].

2 Preliminaries

For a positive integer p we write $[p] = \{1, \ldots, p\}$. We use the $\mathcal{O}_d(\cdot)$ notation to hide multiplicative factors depending on d.

Matrices, divisions, and zones. A *binary matrix* is a matrix with entries in $\{0, 1\}$; all matrices in this paper are binary unless explicitly stated.

Let M be a matrix. Note that rows of M are totally ordered, and similarly for columns of M. A *row block* in M is a non-empty set of rows of M that are consecutive in this total order; *column blocks* are defined analogously. If R is a row block and C is a column block, then the *zone* induced by R and C is the rectangular submatrix of M consisting of entries at the intersections of rows from R and columns from C. In general, by a *submatrix* of M we mean the zone induced by some row block and some column block.

A submatrix is *constant* if all its entries are the same. It is horizontal if all its columns are the same (equivalently, all rows are constant), and *vertical* if all its rows are the same (equivalently, all columns are constant). Note that thus, a constant submatrix is both horizontal and vertical. A submatrix that is neither horizontal nor vertical is called *mixed.*

A *division* of matrix M is a pair $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R}$ is a partition of rows into row blocks and $\mathcal{C}$ is a partition of columns into column blocks. Note that such a division partitions M into $|\mathcal{R}| \cdot |\mathcal{C}|$ zones, each induced by a pair of blocks $(R, C) \in \mathcal{R} \times \mathcal{C}$. We call them the *zones* of the division $(\mathcal{R}, \mathcal{C})$. A *$t$-division* is a division where $|\mathcal{R}| = |\mathcal{C}| = t$.

Twin-width. Let M be a matrix. If $(\mathcal{R}, \mathcal{C})$ is a division of M, then a *contraction* of $(\mathcal{R}, \mathcal{C})$ is any division $(\mathcal{R}', \mathcal{C}')$ obtained from $(\mathcal{R}, \mathcal{C})$ by either merging two consecutive row blocks $R_1, R_2 \in \mathcal{R}$ into a single row block $R_1 \cup R_2$, or merging two consecutive column blocks $C_1, C_2 \in \mathcal{C}$ into a single column block $C_1 \cup C_2$. A *contraction sequence* for M is a sequence of divisions

$$(\mathcal{R}_0, \mathcal{C}_0), (\mathcal{R}_1, \mathcal{C}_1), \ldots, (\mathcal{R}_p, \mathcal{C}_p),$$

such that

- $(\mathcal{R}_0, \mathcal{C}_0)$ is the finest division where every row and every column is in a separate block;
- $(\mathcal{R}_p, \mathcal{C}_p)$ is the coarsest partition where all rows are in a single row block and all columns are in a single column block; and
- for each $i \in \{1, \ldots, p\}$, $(\mathcal{R}_i, \mathcal{C}_i)$ is a contraction of $\mathcal{R}_{i-1}, \mathcal{C}_{i-1}$.

Note that thus, p has to be equal to the sum of the dimension of M minus 2.

Finally, for a division $(\mathcal{R}, \mathcal{C})$ of M, the *error value* of $(\mathcal{R}, \mathcal{C})$ is the least d such that in $(\mathcal{R}, \mathcal{C})$, every row block and every column block contains at most d non-constant zones.

With all these ingredients in place, we can formally define the twin-width of matrices. There are a few alternative definitions spanning through [1, 2, 3, 5]; here we follow the terminology from [5].

▶ **Definition 2.** *A binary matrix M is d-*twin-ordered *if it admits a contraction sequence in which every division has error value at most d. The* twin-width *of a binary matrix M is the least d such that one can permute the rows and columns of M so that the obtained matrix is d-twin-ordered.*

It is straightforward to see that the definition above is equivalent to the one given in the first paragraph of Section 1.

Observe that in the above definition, certifying that a matrix is d-twin-ordered requires showing a suitable contraction sequence where all divisions have error value at most d. In our algorithmic results we do not require that such a contraction sequence is given on input, as we will exploit the assumption that the matrix is d-twin-ordered only through combinatorial properties provided by the connections with mixed minors, which we discuss next. In fact, as discussed [5], it is currently unknown how to efficiently compute a contraction sequence witnessing that a matrix is d-twin-ordered.

Matrix minors and Marcus-Tardos Theorem. We need the following definitions of matrix minors, which intuitively are "complicated substructures" in matrices.

▶ **Definition 3.** *Let M be a binary matrix. A t-*grid minor *in M is a t-division of M where every zone contains at least one entry 1. A t-*mixed minor *in M is a t-division of M where every zone is mixed. We say that M is t-*mixed-free *if M does not contain a t-mixed minor.*

The celebrated result of Marcus and Tardos asserts that if a matrix has a large density of entries 1, then it contains a large grid minor.

▶ **Theorem 4** ([12])**.** *For every $t \in \mathbb{N}$ there exists $c_t \in \mathbb{N}$ such that the following holds. Suppose M is an $n \times m$ binary matrix with at least $c_t \cdot \max(n, m)$ entries 1. Then M has a t-grid minor.*

The currently best upper bound on c_t is $\frac{8}{3}(t+1)^2 2^{4t}$, due to Cibulka and Kynčl [8]. From now on we adopt the constant c_t in the notation.

In [5], Bonnet et al. used the result of Marcus and Tardos to show that, intuitively, large mixed minors are canonical obstacles for having bounded twin-width.

▶ **Theorem 5** ([5])**.** *Let M be a binary matrix. Then the following implications hold:*

- *If M is d-twin-ordered, then M is $(2d+2)$-mixed-free.*
- *If M is t-mixed-free, then M has twin-width at most k_t, where k_t is a constant depending only on t.*

Note that the conclusion of the second implication of Theorem 5 is only a bound on the twin-width: the matrix might still need to be permuted to be k_t-twin-ordered. In this work we will only rely on the first implication of Theorem 5: being d-twin-ordered implies $(2d+2)$-mixed-freeness.

We now derive some simple properties of mixed-free matrices that will be used throughout the paper. First, in a t-mixed-free matrix every ℓ-division has only $\mathcal{O}_t(\ell)$ mixed zones.

▶ **Lemma 6.** *Let M be a t-mixed free matrix, and let $(\mathcal{R}, \mathcal{C})$ be an ℓ-division of M, for some integer ℓ. Then $(\mathcal{R}, \mathcal{C})$ has at most $c_t \cdot \ell$ mixed zones.*

Proof. Construct an $\ell \times \ell$ matrix A by taking the division $(\mathcal{R}, \mathcal{C})$ and substituting each mixed zone with a single entry 1, and each non-mixed zone with a single entry 0. Observe that A may have at most $c_t \cdot \ell$ entries 1, for otherwise, by Theorem 4, A would contain a t-grid minor, which would correspond to a t-mixed minor in M. Hence $(\mathcal{R}, \mathcal{C})$ may have at most $c_t \cdot \ell$ mixed zones. ◀

For the next observations we need the following notion. A *corner* in a matrix M is simply a mixed 2×2 submatrix which is an intersection of two consecutive rows with two consecutive columns. The following observation was pivotally used in the proof of Theorem 5 in [5].

▶ **Lemma 7** ([5])**.** *A matrix is mixed if and only if it contains a corner.*

The next lemma is essentially proven in [5] but never stated explicitly. So we include a proof for completeness.

▶ **Lemma 8** (implicit in [5])**.** *A t-mixed-free $n \times n$ matrix contains at most $2c_t(n+2)$ corners.*

Proof. Let M be a t-mixed-free $n \times n$ matrix. Consider the $\lceil n/2 \rceil$-division $(\mathcal{R}, \mathcal{C})$ of M, in which every row block consists of rows with indices $2i-1$ and $2i$ for some $i \in \{1, \ldots, \lfloor n/2 \rfloor\}$, possibly except the last block that consists only of row n in case n is odd, and similarly for column blocks. By Lemma 6, $(\mathcal{R},\mathcal{C})$ has at most $c_t\lceil n/2\rceil \leq c_t(n/2+1)$ mixed zones, which implies that M has at most $c_t(n/2+1)$ corners in which the bottom-right entry is in the intersection of an even-indexed row and an even-indexed column. Call such corners of *type* 00; corners of types 01, 10, and 11 are defined analogously. By suitably modifying the pairing of rows and columns in $(\mathcal{R},\mathcal{C})$, we can analogously prove that the number of corners of each of the remaining three types is also bounded by $c_t(n/2+1)$. Hence, in total there are at most $4c_t(n/2+1) = 2c_t(n+2)$ corners in M. ◀

Next, we will need a variant of Lemma 6 that focuses on mixed borders between neighboring zones. Here, two different zones in a division $(\mathcal{R},\mathcal{C})$ are called *adjacent* if they are either in the same row block and consecutive column blocks, or in the same column block and consecutive row blocks. A *mixed cut* in $(\mathcal{R},\mathcal{C})$ is a pair of adjacent zones such that there is a corner that crosses the boundary between them, i.e., has two entries in each of them. A *split corner* in $(\mathcal{R},\mathcal{C})$ is a corner intersecting four different zones, i.e., it has an entry in four different zones.

The proof of the following observation is again essentially present in [5].

▶ **Lemma 9.** *Let M be a t-mixed free matrix, and let $(\mathcal{R},\mathcal{C})$ be an ℓ-division of M, for some integer ℓ. Then $(\mathcal{R},\mathcal{C})$ has at most $c_t\cdot(\ell+2)$ mixed cuts and at most $2c_t\cdot(\ell+1)$ split corners.*

Proof. Let $(\mathcal{R}^{00},\mathcal{C}^{00})$ be the division obtained from $(\mathcal{R},\mathcal{C})$ by merging the row blocks indexed $2i-1$ and $2i$ into a single row block, and merging the column blocks indexed $2i-1$ and $2i$ into a single column block, for each $i \in \{1,\ldots,\lfloor \ell/2\rfloor\}$. Obtain divisions $(\mathcal{R}^{10},\mathcal{C}^{10})$, $(\mathcal{R}^{01},\mathcal{C}^{01})$, and $(\mathcal{R}^{11},\mathcal{C}^{11})$ in a similar manner, where if the first number in the superscript is 1 then we merge row blocks $2i$ and $2i+1$ for each $i\in\{1,\ldots,\lceil \ell/2\rceil-1\}$ instead, and if the second number in the superscript is 1 then we merge column blocks $2i$ and $2i+1$ for each $i\in\{1,\ldots,\lceil \ell/2\rceil-1\}$ instead.

Observe that for every mixed cut of $(\mathcal{R},\mathcal{C})$, the two zones in the mixed cut end up in the same zone in either $(\mathcal{R}^{00},\mathcal{C}^{00})$ or in $(\mathcal{R}^{11},\mathcal{C}^{11})$, rendering this zone mixed. However, by Lemma 6, $(\mathcal{R}^{00},\mathcal{C}^{00})$ and $(\mathcal{R}^{11},\mathcal{C}^{11})$ have at most $c_t\cdot(\ell/2+1)$ mixed zones. It follows that $(\mathcal{R},\mathcal{C})$ has at most $2c_t\cdot(\ell/2+1) = c_t\cdot(\ell+2)$ mixed cuts. The bound on the number of split corners follows from the same argument combined with the observation that every split corner in $(\mathcal{R},\mathcal{C})$ is entirely contained in a single zone of exactly one of divisions $(\mathcal{R}^{00},\mathcal{C}^{00})$, $(\mathcal{R}^{10},\mathcal{C}^{10})$, $(\mathcal{R}^{01},\mathcal{C}^{01})$, and $(\mathcal{R}^{11},\mathcal{C}^{11})$. ◀

Partitioning into rectangles. We conclude with another observation about twin-ordered matrices: they can be decomposed into a small number of rectangles. Formally, for a binary matrix M, a *rectangle decomposition* of M is a set $\mathcal{K}$ of pairwise disjoint rectangular submatrices (i.e., zones induced by some row block and some column block) such that every submatrix in $\mathcal{K}$ is entirely filled with 1s and there is no entry 1 outside the submatrices in $\mathcal{K}$. The following lemma is stated and proved in the graph setting in [2]; we adapt the proof here to the matrix setting.

▶ **Lemma 10.** *Let M be an $n\times n$ binary matrix that is d-twin-ordered. Then M admits a rectangle decomposition $\mathcal{K}$ with $|\mathcal{K}|\leq d(2n-2)+1$.*

Proof. Let $(\mathcal{R}_0,\mathcal{C}_0),\ldots,(\mathcal{R}_{2n-2},\mathcal{C}_{2n-2})$ be a contraction sequence for M with error value at most d. Let $\mathcal{S}_i$ be the set of zones of $(\mathcal{R}_i,\mathcal{C}_i)$, and let

$$\mathcal{S} = \bigcup_{i=0}^{2n-2} \mathcal{S}_i.$$

Note that $\mathcal{S}$ is a *laminar* family, that is, every two submatrices in $\mathcal{S}$ are either disjoint or one is contained in the other.

Let $\mathcal{K}$ be the subfamily of $\mathcal{S}$ consisting of those submatrices that are entirely filled with 1s, and are inclusion-wise maximal in $\mathcal{S}$ subject to this property. Note that every entry 1 in M is contained in some member of $\mathcal{K}$, for the zone of $(\mathcal{R}_0, \mathcal{C}_0)$ in which this entry is contained is a 1×1 submatrix entirely filled with 1s. Since $\mathcal{S}$ is laminar, it follows that $\mathcal{K}$ is a rectangle decomposition of M. So it remains to argue that $|\mathcal{K}| \le d(2n-2)+1$.

Consider any $A \in \mathcal{K}$ and let i be the largest index such that $A \in \mathcal{S}_i$. We may assume that $i < 2n-2$, for otherwise the matrix M is entirely filled with 1s and the postulated claim is trivial. By maximality, A is contained in a non-constant zone $B \in \mathcal{S}_{i+1}$ that resulted from merging A with another adjacent zone $A' \in \mathcal{S}_i$, which is not entirely filled with 1s. In particular, B lies in the unique row block or column block of $(\mathcal{R}_{i+1}, \mathcal{C}_{i+1})$ that resulted from merging two row blocks or two column blocks of $(\mathcal{R}_i, \mathcal{C}_i)$. There can be at most d non-constant zones in this row/column block of $(\mathcal{R}_{i+1}, \mathcal{C}_{i+1})$, and B is one of them. We infer that i can be the largest index satisfying $A \in \mathcal{S}_i$ for at most d different submatrices $A \in \mathcal{K}$. Since this applies to every index $i \in \{0, 1, \ldots, 2n-3\}$, we conclude that $|\mathcal{K}| \le d(2n-2)$. ◀

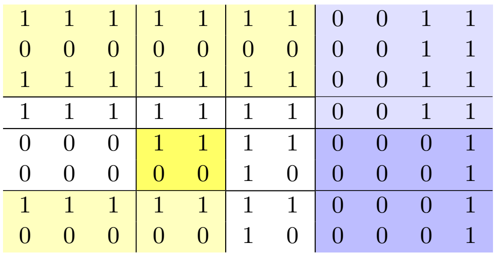
Observe that Lemma 10 provides a way to encode an $n \times n$ d-twin-ordered matrix in $\mathcal{O}_d(n \log n)$ bits: one only needs to specify the vertices of the submatrices of a rectangle decomposition of size $\mathcal{O}_d(n)$. The proof is also effective, in the sense that given a suitably represented contraction sequence one can compute the obtained decomposition $\mathcal{K}$. To abstract away the nuances of representing contraction sequences, throughout this paper we assume that d-twin-ordered matrices are provided on input through suitable rectangle decompositions.

3 Structural properties of divisions

Before we proceed to constructing the promised compact representation, we need to describe some new combinatorial properties of twin-ordered matrices. For the remainder of this section, we fix $d \in \mathbb{N}$ and consider a matrix M that is d-twin-ordered. In particular, by Theorem 5, M is $(2d+2)$-mixed-free.

Strips. We begin by considering non-constant vertical and horizontal zones of a given division of M. We will show that these zones can be grouped into $\mathcal{O}_d(t)$ *strips* that again are vertical or horizontal, respectively. This partitioning is formalized as follows.

▶ **Definition 11.** *Let $(\mathcal{R}, \mathcal{C})$ be a division of a matrix M. A* vertical strip *in $(\mathcal{R}, \mathcal{C})$ is an inclusion-wise maximal set of non-constant vertical zones of $\mathcal{D}$ that are contained in the same column block of $(\mathcal{R}, \mathcal{C})$, span a contiguous interval of row blocks, and whose union is again a vertical submatrix.* Horizontal strips *are defined analogously.*



1	1	1	1	1	1	1	0	0	1	1
0	0	0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	0	0	1	1
0	0	0	1	1	1	1	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1
1	1	1	1	1	1	1	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1

Figure 1 Strips in an example 4-division of a matrix. Horizontal strips are painted in shades of yellow. Vertical strips are painted in shades of blue. Unpainted zones are constant or mixed.

Naturally, each non-constant vertical zone belongs to exactly one vertical strip; and similarly, each non-constant horizontal zone belongs to exactly one horizontal strip.

We will now show an upper bound on the number of vertical and horizontal strips present in any t-division of M.

▶ **Lemma 12.** *For every $t \in \mathbb{N}$, the total number of vertical and horizontal strips in any t-division of M is at most $\mathcal{O}_d(t)$.*

Proof. We focus on the bound for vertical strips only; the proof for horizontal strips is symmetric. Fix some t-division $(\mathcal{R}, \mathcal{C})$ of M. Observe that each vertical strip S of the division either intersects the top row of the matrix, or the top-most zone of S is adjacent from the top to another zone C such that adding C to S yields a submatrix that is not vertical. (We say that C is adjacent to S *from the top.*) Thus, we partition the family of vertical strips in the t-division of M into three types:

(I) strips intersecting the top row of M;
(II) strips adjacent to a mixed zone C from the top; and
(III) strips adjacent to a non-mixed zone C from the top.

Obviously, there are at most t vertical strips of type (I). Next, each vertical strip of type (II) can be assigned a private mixed zone C adjacent to it from the top. Hence, the number of vertical strips of this type is upper bounded by the number of mixed zones in $(\mathcal{R}, \mathcal{C})$, which by Lemma 6 is bounded by $\mathcal{O}_d(t)$.

Finally, let us consider vertical strips of type (III). Let S be a vertical strip of this type, D be its top-most zone, and C be the non-mixed zone adjacent to D from the top. Since D is vertical, all rows of D are repetitions of the same row vector v_D. Since D is non-constant, v_D is non-constant as well.

As C is non-mixed, it is either horizontal or vertical. If C is vertical, then all its rows are repetitions of the same row vector v_C. Observe that since strip S could not be extended by C, we have $v_C \neq v_D$. Now, as v_D is non-constant, it follows that the union of the bottom-most row of C and the top-most row of D contains a corner. On the other hand, if C is horizontal, then the bottom-most row of C is constant and again there is a corner in the union of the (constant) bottom-most row of C and the (non-constant) top-most row of D.

So in both cases we conclude that C and D form a mixed cut. By Lemma 9, the total number of mixed cuts in $(\mathcal{R}, \mathcal{C})$ is bounded by $\mathcal{O}_d(t)$, so also there are at most $\mathcal{O}_d(t)$ vertical strips of type (III). This concludes the proof. ◀

Regular divisions. We move our focus to a central notion of our data structure: *regular divisions* of a matrix:

▶ **Definition 13.** *Given M and an integer $s \in \mathbb{N}$, we define the s-*regular division *of M as the $\lceil \frac{n}{s} \rceil$-division of M in which each row block (respectively, column block), possibly except the last one, contains s rows (resp. columns). Precisely, if $s \nmid n$, then the last row block and the last column block contain exactly $n \bmod s$ rows or columns, respectively.*

In the data structure, given a square input matrix M, we will construct multiple regular divisions of M of varying granularity (the value of s). Crucially, in order to ensure the space efficiency of the data structure, we will require that the number of *distinct* zones in each such regular division of M should be small. This is facilitated by the following definition:

▶ **Definition 14.** *For $s \in \mathbb{N}$, the s-*zone family *of M, denoted $\mathcal{F}_s(M)$, is the set of all different zones participating in the s-regular division of M.*

Let us stress that we treat $\mathcal{F}_s(M)$ as a set of matrices and do not keep duplicates in it. That is, if the regular s-division of M contains two or more isomorphic zones – with same dimensions and equal corresponding entries – then these zones are represented in $\mathcal{F}_s(M)$ only once.

For the remainder of this section, we will prove good bounds on the cardinality of $\mathcal{F}_s(M)$. Trivially, the cardinality of $\mathcal{F}_s(M)$ is bounded by $\left\lceil \frac{n}{s} \right\rceil^2$ (i.e., the number of zones in the s-regular division). Also, the same cardinality is trivially bounded by $2^{\mathcal{O}(s^2)}$ (i.e., the total number of distinct matrices with at most s rows and columns). However, given that M is d-twin-ordered, both bounds can be improved dramatically. First, the dependence on $\frac{n}{s}$ in the former bound can be improved to linear:

▶ **Lemma 15.** *For every $s \in \{1, \dots, n\}$, the cardinality of $\mathcal{F}_s(M)$ is bounded by $\mathcal{O}_d(\frac{n}{s})$.*

Proof. First assume that $s \mid n$; hence, each zone in the s-regular division of M has s rows and s columns. Then, the matrices in $\mathcal{F}_s(M)$ can be categorized into four types:

- Constant zones. There are at most 2 of them – constant 0 and constant 1.
- Mixed zones. Here, Lemma 6 applies directly: since the considered division is an $\frac{n}{s}$-division of M, there are at most $\mathcal{O}_d(\frac{n}{s})$ mixed zones in M in total.
- Vertical zones. By Lemma 12, all vertical zones of the considered division can be partitioned into $\mathcal{O}_d(\frac{n}{s})$ vertical strips. As all zones have the same dimensions, the zones belonging to a single vertical strip are pairwise isomorphic. From this we infer the $\mathcal{O}_d(\frac{n}{s})$ upper bound on the number of different vertical zones.
- Horizontal zones are handled symmetrically to vertical zones.

Finally, if $s \nmid n$, then let M' be equal to M, truncated to the first $n - (n \bmod s)$ rows and columns; equivalently, M' is equal to M with all zones with fewer than s rows or columns removed. The argument given above applies to M', yielding at most $\mathcal{O}_d(\frac{n}{s})$ different $s \times s$ zones in M' (and equivalently in M). The proof is concluded by the observation that M contains exactly $2\left\lceil \frac{n}{s} \right\rceil - 1 = \mathcal{O}(\frac{n}{s})$ zones in its s-regular division that have fewer than s rows or columns. ◀

Second, from the works of Bonnet et al. [1, 3] one can easily derive an upper bound that is exponential in s rather than in s^2:

▶ **Lemma 16.** *For every $s \in \{1, \dots, n\}$, the cardinality of $\mathcal{F}_s(M)$ is bounded by $2^{\mathcal{O}_d(s)}$.*

Proof. Observe that a submatrix of a d-twin-ordered matrix is also d-twin-ordered. Thus, it is only necessary to upper bound the total number of different $s \times s$ matrices that are d-twin-ordered. To this end, we use the notion of twin-width of ordered binary relational structures introduced in the work of Bonnet et al. [3]. This notion is more general than twin-orderedness in the following sense: each $s \times s$ matrix that is d-twin-ordered corresponds to a different ordered binary structure over s elements of twin-width at most d. As proved in [3], the number of different such structures is upper bounded by $2^{\mathcal{O}_d(s)}$. The claim follows. ◀

While the bound postulated by Lemma 15 is more powerful for coarse regular divisions of M (i.e., s-regular divisions for large s), Lemma 16 yields a better bound for $s \leq p_d \cdot \log n$, where $p_d > 0$ is a sufficiently small constant depending on d.

4 Data structure

In this section we present the data structure promised in Theorem 1. Recall that it should represent a given binary $n \times n$ matrix M that is d-twin-ordered, and it should provide access to the following query: for given $(i, j) \in [n]^2$, return the entry $M[i, j]$. Here we focus only on the description of the data structure, implementation of the query, and analysis of the bitsize. The construction algorithm promised in Theorem 1 is given in the full version of the work [14].

Without loss of generality, we assume that n is a power of 2. Otherwise we enlarge M, so that its order is the smallest power of 2 larger than n. We use dummy 0s to fill additional entries. It is straightforward to see that the resulting matrix is $(d+1)$-twin-ordered. Similarly, in the analysis we may assume that n is sufficiently large compared to any constants present in the context.

Description. Our data structure consists of $\ell+1$ layers: $\mathcal{L}_0, \ldots, \mathcal{L}_\ell$. Recall from Definition 14 that $\mathcal{F}_s(M)$ is the family of pairwise different zones participating in the s-regular division of M. Each layer $\mathcal{L}_i$ in our data structure corresponds to $\mathcal{F}_{m_i}(M)$ for a carefully chosen parameter m_i. Let $\mathsf{low}(x)$ be the largest power of 2 smaller or equal to x. We define parameters m_i inductively as follows: set $m_0 = n$ and for $i \geq 0$,

$$m_{i+1} = \begin{cases} \mathsf{low}({m_i}^{2/3}) & \text{if } m_i \geq \log^3 n \\ m_i/2 & \text{if } \log n/(2\beta_d) \leq m_i < \log^3 n \end{cases}$$

where β_d is the constant hidden in the $\mathcal{O}_d(\cdot)$ notation in Lemma 16, i.e., $|\mathcal{F}_s(M)| \leq 2^{\beta_d \cdot s}$. The construction stops when we reach m_i satisfying $m_i < \log n/(2\beta_d)$, in which case we set $\ell = i$. Note that all parameters m_i are powers of 2, so m_j divides m_i whenever $i \leq j$.

We also observe the following.

▷ Claim 17. $\ell \in \mathcal{O}(\log \log n)$.

Proof. Let k be the least index for which $m_k < \log^3 n$. Observe that for $i \in [1, k]$ we have $m_i \leq n^{(2/3)^i}$. So it must be that $k \leq \log_{3/2} \log n + 1 \in \mathcal{O}(\log \log n)$, for otherwise we would have $m_{k-1} \leq n^{(2/3)^{\log_{3/2} \log n}} = n^{1/\log n} = 2 < \log^3 n$. Next, observe that for $i \in [k+1, \ell]$ we have $m_i = m_k/2^{i-k}$. Therefore, we must have $\ell - k \leq \log(\log^3 n) + 1 \in \mathcal{O}(\log \log n)$, for otherwise we have $m_{\ell-1} \leq m_k/2^{\log \log^3 n} < \log^3 n/\log^3 n = 1$. The claim follows. ◁

Layer $\mathcal{L}_\ell$ is special and we describe it separately, so let us now describe the content of layer $\mathcal{L}_i$ for each $i < \ell$. Since n is divisible by m_i, every $Z \in \mathcal{F}_{m_i}(M)$ is an $m_i \times m_i$ matrix that appears at least once as a zone in the (n/m_i)-regular division of M. Such Z will be represented by an object $\mathsf{obj}(Z)$ in $\mathcal{L}_i$. Each object $\mathsf{obj}(Z)$ stores $(m_i/m_{i+1})^2$ pointers to objects in $\mathcal{L}_{i+1}$; recall here that m_{i+1} divides m_i. Consider the m_{i+1}-regular division of Z. This division consists of $(m_i/m_{i+1})^2$ zones; index them as $\mathsf{subzone}_Z(i,j)$ for $i, j \in [m_i/m_{i+1}]$ naturally. Observe that for all $i, j \in [m_i/m_{i+1}]$, it holds that $\mathsf{subzone}_Z(i,j) \in \mathcal{F}_{m_{i+1}}(M)$. In our data structure, each object $\mathsf{obj}(Z) \in \mathcal{L}_i$, corresponding to a matrix $Z \in \mathcal{F}_{m_i}(M)$, stores an array ptr of $(m_i/m_{i+1})^2$ pointers, where $\mathsf{ptr}[i,j]$ points to the address of $\mathsf{subzone}_Z(i,j)$ for all $i, j \in [m_i/m_{i+1}]$. This concludes the description of layer $\mathcal{L}_i$ for $i < \ell$.

We now describe layer $\mathcal{L}_\ell$. It is also a collection of objects, and for each matrix $Z \in \mathcal{F}_{m_\ell}(M)$ there is an object $\mathsf{obj}(Z) \in \mathcal{L}_\ell$; these objects are pointed to by objects from $\mathcal{L}_{\ell-1}$. However, instead of storing further pointers, each object $\mathsf{obj}(Z) \in \mathcal{L}_\ell$ stores the entire matrix $Z \in \mathcal{F}_{m_\ell}(M)$ as a binary matrix of order $m_\ell \times m_\ell$, using m_ℓ^2 bits. This concludes the description of $\mathcal{L}_\ell$.

Observe that in $\mathcal{L}_0$ there is only one object corresponding to the entire matrix M. We store a global pointer ptrGlo to this object. Our data structure is accessed via ptrGlo upon each query.

Implementation of the query. The description of the data structure is now complete and we move on to describing how the query is executed. The query is implemented as method $\mathsf{entry}(i,j)$ and returns $M[i,j]$; see Algorithm 1 for the pseudocode (where $\mathsf{ptrIt} \to$ stands for dereference of a pointer ptrIt, i.e., the object pointed to by ptrIt). Given two integers $i,j \in [n]$, the method starts with pointer ptrGlo, and uses i and j and iterator pointer ptrIt to navigate via pointers down the layers, ending with a pointer to an object in layer $\mathcal{L}_\ell$. Initially, the iterator ptrIt is set to ptrGlo and it points to $\mathsf{obj}(Z)$ for the only matrix $Z \in \mathcal{F}_{m_0}(M)$. Integers i,j are the positions of the desired entry with respect to zone Z. After a number of iterations, ptrIt points to an object $\mathsf{obj}(Z) \in \mathcal{L}_k$ for a matrix $Z \in \mathcal{F}_{m_k}(M)$, and maintains current coordinates i and j. The invariant is that the desired output is the entry $Z[i,j]$. In one step of the iteration, the algorithm finds the matrix Z' in $\mathcal{F}_{m_{k+1}}(M)$ containing the desired entry $Z[i,j]$, which is the zone $\mathsf{subzone}_Z(i \mathbin{\mathsf{div}} m_{k+1}, j \mathbin{\mathsf{div}} m_{k+1}) \in \mathcal{F}_{m_{k+1}}(M)$, and moves the pointer ptrIt to $\mathsf{obj}(Z') \in \mathcal{L}_{k+1}$. The new coordinates of the desired entry with respect to Z' are $(i \mathbin{\mathsf{mod}} m_{k+1})$ and $(j \mathbin{\mathsf{mod}} m_{k+1})$, so i and j are altered accordingly. Once the iteration reaches $\mathcal{L}_\ell$, the object pointed to by ptrIt contains the entire zone explicitly, so it suffices to return the desired entry. Obviously, the running time of the query is $\mathcal{O}(\log\log n)$, since the algorithm iterates through $\ell \in \mathcal{O}(\log\log n)$ layers.

Algorithm 1 Query algorithm.

```
Input  : Integers $i, j \in [n]$
Output : $M[i,j]$
ptrIt ← ptrGlo
for $k \leftarrow 0$ to $\ell - 1$ do
    ptrIt ← (ptrIt → ptr[$i$ div $m_{k+1}$, $j$ div $m_{k+1}$] ;
    $i \leftarrow i$ mod $m_{k+1}$ ;
    $j \leftarrow j$ mod $m_{k+1}$ ;
return ptrIt → $Z[i,j]$
```

Analysis of bitsize. We now analyze the number of bits occupied by the data structure. First note that the total number of objects stored is bounded by the total number of submatrices of M, which is polynomial in n. Hence, every pointer can be represented using $\mathcal{O}(\log n)$ bits. Keeping this in mind, the total bitsize occupied by the data structure is proportional to

$$\sum_{i=0}^{\ell-1} |\mathcal{F}_{m_i}(M)| \left(\frac{m_i}{m_{i+1}}\right)^2 \log n + |\mathcal{F}_{m_\ell}(M)| m_\ell^2, \tag{1}$$

This is because for all layers $\mathcal{L}_i$ for $i < \ell$ we store $|\mathcal{F}_{m_i}(M)|$ objects, each storing $\left(\frac{m_i}{m_{i+1}}\right)^2$ pointers, and in $\mathcal{L}_\ell$ we store $|\mathcal{F}_{m_\ell}(M)|$ objects, each storing a binary matrix of order $m_\ell \times m_\ell$.

We first bound the second term of Equation (1). By Lemma 16, we have

$$|\mathcal{F}_{m_\ell}(M)| m_\ell^2 \leq 2^{\beta_d \cdot m_\ell} \cdot m_\ell^2 \leq 2^{\beta_d \cdot \frac{\log n}{2\beta_d}} \cdot \left(\frac{\log n}{2\beta_d}\right)^2 = \sqrt{n} \cdot \left(\frac{\log n}{2\beta_d}\right)^2 \in o(n).$$

We move on to bounding the first term of Equation (1). Let k be the least index for which $m_k < \log^3 n$. We can split the first term of Equation (1) into two sums:

$$\sum_{i=0}^{\ell-1} |\mathcal{F}_{m_i}(M)| \left(\frac{m_i}{m_{i+1}}\right)^2 \log n =$$

$$= \sum_{i=0}^{k-1} |\mathcal{F}_{m_i}(M)| \left(\frac{m_i}{m_{i+1}}\right)^2 \log n + \tag{2}$$

$$+ \sum_{i=k}^{\ell-1} |\mathcal{F}_{m_i}(M)| \left(\frac{m_i}{m_{i+1}}\right)^2 \log n \tag{3}$$

We first apply Lemma 15 to bound the sum (2). More precisely, if α_d is the constant hidden in the $\mathcal{O}_d(\cdot)$ notation in Lemma 15, we have

$$(2) \leq \log n \cdot \sum_{i=0}^{k-1} \alpha_d \frac{n}{m_i} \cdot 4m_i^{2/3} = 4\alpha_d n \log n \cdot \sum_{i=0}^{k-1} \frac{1}{m_i^{1/3}} \tag{4}$$

Since for $i \in [k-1]$ we have $m_{i+1} = \mathsf{low}(m_i^{2/3})$ and $m_i \geq \log^3 n$, we have $m_i/m_{i+1} \geq 2$. Therefore $m_i \geq 2^{k-i-1} m_{k-1}$ for $i \in [0, k-1]$, so we can continue bounding the last expression in Equation (4):

$$(4) \leq \alpha_d n \log n \sum_{i=0}^{k-1} \cdot \frac{1}{(2^{k-i-1} m_{k-1})^{1/3}} \leq \alpha_d n \cdot \frac{\log n}{m_{k-1}^{1/3}} \cdot \sum_{i=0}^{k-1} \frac{1}{(2^{k-i-1})^{1/3}} \in \mathcal{O}_d(n).$$

It remains to bound sum (3). We use Lemma 15 similarly as above:

$$(3) = 4 \log n \cdot \sum_{i=k}^{\ell-1} |\mathcal{F}_{m_i}(M)| \leq 4\alpha_d n \log n \cdot \sum_{i=k}^{\ell-1} \frac{1}{m_i} \leq 4\alpha_d n \log n \cdot \sum_{i=0}^{\infty} \frac{1}{(\frac{\log n}{2\beta_d}) \cdot 2^i} \in \mathcal{O}_d(n).$$

By summing up all the bounds we infer that the total number of bits occupied by our data structure is $\mathcal{O}_d(n)$.

References

1 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In *2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1977–1996. SIAM, 2021. doi:10.1137/1.9781611976465.118.

2 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: Max Independent Set, Min Dominating Set, and Coloring. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 35:1–35:20. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2021.

3 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width IV: ordered graphs and matrices. *CoRR*, abs/2102.03117, 2021. arXiv:2102.03117.

4 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. Twin-width and polynomial kernels. *CoRR*, abs/2107.02882, 2021. arXiv:2107.02882.

5 Édouard Bonnet, Eun Jung Kim, Stéphan Thomasse, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *IEEE 61st Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 601–612. IEEE Computer Society, 2020.

6 Édouard Bonnet, Jaroslav Nešetřil, Patrice Ossona de Mendez, Sebastian Siebertz, and Stéphan Thomassé. Twin-width and permutations. *CoRR*, abs/2102.06880, 2021. arXiv:2102.06880.

7 Timothy M. Chan. Persistent Predecessor Search and Orthogonal Point Location on the Word RAM. *ACM Trans. Algorithms*, 9(3):22:1–22:22, 2013.

8 Josef Cibulka and Jan Kynčl. Better upper bounds on the Füredi-Hajnal limits of permutations, 2019. arXiv:1607.07491.

9 Jan Dreier, Jakub Gajarský, Yiting Jiang, Patrice Ossona de Mendez, and Jean-Florent Raymond. Twin-width and generalized coloring numbers. *CoRR*, abs/2104.09360, 2021. arXiv:2104.09360.

10 Jakub Gajarský, Michał Pilipczuk, and Szymon Toruńczyk. Stable graphs of bounded twin-width. *CoRR*, abs/2107.03711, 2021. arXiv:2107.03711.

11 Shahin Kamali. Compact representation of graphs of small clique-width. *Algorithmica*, 80(7):2106–2131, 2018.

12 Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley–Wilf conjecture. *Journal of Combinatorial Theory, Series A*, 107(1):153–160, 2004.

13 Mihai Pătraşcu and Mikkel Thorup. Time-space trade-offs for predecessor search. In *38th Annual ACM Symposium on Theory of Computing, STOC 2006*, pages 232–240. ACM, 2006.

14 Michał Pilipczuk, Marek Sokołowski, and Anna Zych-Pawlewicz. Compact representation for matrices of bounded twin-width. *CoRR*, abs/2110.08106, 2021. arXiv:2110.08106.

A Representation with bitsize $\mathcal{O}(n^{1+\varepsilon})$ and query time $\mathcal{O}(1/\varepsilon)$

In this section we provide a brief sketch of another data structure representing twin-ordered matrices. For any fixed $\varepsilon > 0$, we will construct a data structure that represents a given d-twin-ordered $n \times n$ matrix M in bitsize $\mathcal{O}(n^{1+\varepsilon})$, and can be queried for entries of M in worst-case time $\mathcal{O}(1/\varepsilon)$ per query.

Actually, the data structure solves the Orthogonal Point Location problem. An instance of the problem is a set of $\mathcal{O}(n)$ orthogonal rectangles with pairwise disjoint interiors, each with integer coordinates between 0 and n. In the problem, we are required to preprocess the input rectangles and construct a data structure that can can efficiently locate the rectangle containing a given query point. As the set of 1 entries in any d-twin-ordered matrix M admits a rectangle decomposition into $\mathcal{O}_d(n)$ rectangles (Lemma 10), this also yields a data structure representing M.

Famously, Chan [7] designed a data structure for Orthogonal Point Location that can answer each query in worst-case time $\mathcal{O}(\log\log n)$ and can be constructed in time $\mathcal{O}(n \log\log n)$. In the same work, he also observed that achieving constant query time is much more difficult. Namely, Orthogonal Point Location can be reduced to the static variant of the Predecessor Search problem. Pătraşcu and Thorup proved that each data structure for Predecessor Search with $\mathcal{O}(n \log^{\mathcal{O}(1)} n)$ bitsize necessarily requires $\Omega(\log\log n)$ query time, even in a much more powerful cell probe model [13]. Therefore, for general Orthogonal Point Location, one cannot expect to achieve constant query time with bitsize significantly smaller than $\mathcal{O}(n^{1+\varepsilon})$.

Data structure for disjoint intervals. Consider integers $k, h \geq 1$, and let $n = k^h$. We will first sketch a data structure that maintains a set of disjoint integer intervals that are subintervals of $[0, n-1]$. The data structure shall allow adding or removing intervals in time $\mathcal{O}(kh)$ and querying whether a point is contained in any interval in time $\mathcal{O}(h)$.

Consider a perfect k-ary tree of depth h. The tree has k^h leaves, numbered from 0 to $n-1$ according to the pre-order traversal of the tree. Each internal node at depth $i \in \{0, 1, \ldots, h-1\}$ in the tree corresponds to a contiguous interval of leaves of length k^{h-i}. Each such interval is called a *base interval.* Each internal node contains an array of k pointers to the children in the tree, allowing access to the j-th child in constant time. Additionally, alongside each node v of the data structure, we store an additional bit b_v, initially set to 0.

Assume an interval $[\ell, r]$ is to be inserted to the set. We traverse the tree recursively, starting from the root, entering only nodes whose base intervals intersect $[\ell, r]$, and cutting the recursion at nodes whose base intervals are entirely within $[\ell, r]$. It can be shown that the recursion visits at most $\mathcal{O}(kh)$ nodes and decomposes $[\ell, r]$ into $\mathcal{O}(kh)$ disjoint base intervals. For each node v corresponding to such a base interval, we set $b_v \leftarrow 1$. Removing an interval from the set is analogous. Now, to verify whether an element y belongs to the set, we descend recursively from the root of the tree to the y-th leaf of the tree and verify if any of the visited nodes v has $b_v = 1$. This requires time $\mathcal{O}(h)$.

Since each update and query to the data structure is essentially a recursive search from the root of the tree, the data structure can be made persistent: on each update, we create a copy of each altered node and each of their ancestors, and we reset the pointers in the copies accordingly. As $\mathcal{O}(kh)$ nodes are updated at each query, and each internal node stores an array of $\mathcal{O}(k)$ pointers, the update time increases to $\mathcal{O}(k^2h)$ due to the copying of the nodes; and each update increases the bitsize of the data structure by $\mathcal{O}(k^2h\log n)$. Thus, after $\mathcal{O}_d(n)$ updates, the bitsize of the data structure is $\mathcal{O}_d(nk^2h\log n)$. The query time remains at $\mathcal{O}(h)$.

Orthogonal point location with small coordinates. Fix any $\varepsilon > 0$. Given a matrix M of order n, we set $h := \lceil 2/\varepsilon \rceil + 1$ and $k := \lceil n^{1/h} \rceil$. We instantiate a persistent k-ary tree of depth h as above. We sweep the set of rectangles from the left of the right, maintaining a vertical sweep line. The tree maintains an intersection of the sweep line with the union of rectangles as a set of disjoint intervals contained in $[0, n]$. Hence, for each rectangle, the tree is updated twice: a vertical interval is added when the sweep line reaches the left end of the rectangle, and is removed as soon as it reaches the right end of the rectangle. At each x coordinate, we store the pointer ver_x to the root of the current version of the tree. After the preprocessing, for each query (x, y), we fetch the pointer ver_x and check whether this version of the tree contains y as an element.

Let us analyze the query time and the bitsize of the data structure. For convenience, let $\delta := 1/h$. We can see that $0 < \delta < \frac{\varepsilon}{2}$. Each query is performed in time $\mathcal{O}(h) = \mathcal{O}(1/\varepsilon)$. Storing pointers ver_x requires bitsize $\mathcal{O}(n \log n)$. Since we processed $\mathcal{O}_d(n)$ rectangles, the persistent tree has bitsize $\mathcal{O}_d(nk^2h\log n) = \mathcal{O}_d(n^{1+2\delta}\log n/\varepsilon) = \mathcal{O}_d(n^{1+\varepsilon})$.

On Finer Separations Between Subclasses of Read-Once Oblivious ABPs

C. Ramya ✉ 🏠
Chennai Mathematical Institute, India

Anamay Tengse ✉ 🏠
Department of Computer Science, University of Haifa, Israel

Abstract

Read-once Oblivious Algebraic Branching Programs (ROABPs) compute polynomials as products of univariate polynomials that have matrices as coefficients. In an attempt to understand the landscape of algebraic complexity classes surrounding ROABPs, we study classes of ROABPs based on the algebraic structure of these coefficient matrices. We study connections between polynomials computed by these structured variants of ROABPs and other well-known classes of polynomials (such as depth-three powering circuits, tensor-rank and Waring rank of polynomials).

Our main result concerns *commutative ROABPs*, where *all* coefficient matrices commute with each other, and *diagonal ROABPs*, where all the coefficient matrices are just diagonal matrices. In particular, we show a somewhat surprising connection between these models and the model of *depth-three powering circuits* that is related to the *Waring rank* of polynomials. We show that if the *dimension of partial derivatives* captures *Waring rank* up to polynomial factors, then the model of *diagonal ROABPs* efficiently simulates the seemingly more expressive model of *commutative ROABPs*. Further, a *commutative ROABP* that cannot be efficiently simulated by a *diagonal ROABP* will give an explicit polynomial that gives a super-polynomial separation between *dimension of partial derivatives* and *Waring rank*.

Our proof of the above result builds on the results of Marinari, Möller and Mora (1993), and Möller and Stetter (1995), that characterise rings of commuting matrices in terms of polynomials that have small dimension of partial derivatives. The algebraic structure of the coefficient matrices of these ROABPs plays a crucial role in our proofs.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic Complexity Theory, Algebraic Branching Programs, Commutative Matrices

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.53

Related Version *Full Version*: `https://anamay.bitbucket.io/assets/docs/papers/ROABP-Hierarchy.pdf`

Funding *C. Ramya*: Research supported by INSPIRE Faculty Fellowship of DST and by a grant from the Infosys Foundation. Part of this work was done when at the Tata Institute of Fundamental Research, Mumbai, India (DAE project 12-R&D-TFR-5.01-0500).
Anamay Tengse: Research supported by the Israel Science Foundation (grant No. 716/20). Part of this work was done when at the Tata Institute of Fundamental Research, Mumbai, India, as a student (DAE project no. 12-R&D-TFR-5.01-0500), and later as a visitor (Prof. Prahladh Harsha's Swarnajaynti fellowship and Prof. Arkadev Chattopadhyay's Microsoft Research funds).

Acknowledgements We thank Ramprasad Saptharishi for numerous insightful discussions about the various structured models, which motivated this work. We also thank Mrinal Kumar for his helpful comments about our work which helped us in enchancing the presentation.
We thank Manoj Gopalakrishan and the organisers of *Thursday Theory Lunch* at IIT Bombay for organising a talk by Debasattam Pal, where we first came across the work of Möller and Stetter (1995) that essentially led to the main results in this paper.
We thank the anonymous reviewers for their valuable inputs on the earlier version of the paper.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 53; pp. 53:1–53:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The central question in *algebraic complexity theory*: the theory concerning computation of polynomials, is to understand the most efficient way of computing a polynomial $f(x_1, \ldots, x_n)$ using the basic arithmetic operations of addition and multiplication. One of the earliest works to study the computational complexity of an *explicit* polynomial is perhaps the famous work of Strassen [22] on matrix multiplication. However, the seminal work of Valiant [23] that proposed the "VP vs VNP" question (the algebraic analogue of P vs NP) is widely regarded as the starting point of algebraic complexity theory.

Algebraic circuits are a fundamental model for computing polynomials, and the complexity of a polynomial is determined by the *size* of the smallest circuit that computes it. This definition also coincides with the fewest number of arithmetic operations required to evaluate a polynomial. Valiant's above mentioned work however, uses the model of *algebraic branching programs (ABPs)* to capture *efficiently computable polynomials.* Informally, an ABP computes a polynomial $f(\mathbf{x})$ as the $(1,1)$th entry of a product of matrices, each of which has linear forms in the $\mathbf{x}$ variables as its entries. While VP is the class of n-variate polynomials having poly(n) size algebraic circuits, the class of n-variate polynomials that have an ABP of size poly(n) is called VBP. The class VBP is known to be a subclass of VP, and at the moment it is unclear if this inclusion is strict. The VBP vs VNP question remains a central question in algebraic complexity theory as it is captured by the "determinant vs permanent" question (see e.g. [10]).

Although proving strong lower bounds against algebraic circuits seems currently unattainable, even proving lower bounds against ABPs remains a challenging task. In fact, even a super-quadratic lower bound against ABPs will be a massive improvement over the state of the art ([1, 2]). A significant amount of work in the area has therefore focused on analysing more structured variants of ABPs which could potentially be easier to tackle. Indeed, a celebrated result of Nisan [14] gives an exact characterisation of the complexity of a *non-commutative ABP* computing any non-commutative polynomial[1]. This characterisation yields a $2^{\Omega(n)}$ lower bound against non-commutative ABPs for the determinant, which among other things, highlights the power of commutativity.

We now turn to the protagonists of our work, Read-once Oblivious ABPs (ROABPs), which are the commutative analogues of non-commutative ABPs. ROABPs were first introduced by Forbes and Shpilka [6], in the context of *polynomial identity testing*: another central problem in algebraic complexity, which we discuss in more detail in the full version. An ROABP is an algebraic branching program that uses exactly n matrices, one for each variable; and the entries in the matrix corresponding to an x_i are univariate polynomials from $\mathbb{C}[x_i]$ (formally defined in Definition 9). It is easy to check that ROABPs can compute any monomial, and are closed under taking sums. Thus, every n-variate, degree-d polynomial trivially has an ROABP of size $d^{O(n)}$. On the other hand, Nisan's characterisation [14] for non-commutative ABPs also extends to ROABPs, and hence most of the strong lower bounds against non-commutative ABPs can be suitably translated to ROABPs.

Since all ROABPs use n matrices, the parameter of interest is the *width* of an ROABP, which is the maximum dimension of any of the underlying matrices. Furthermore, since every matrix in an ROABP is associated with exactly one variable in $\{x_1, \ldots, x_n\}$, one can naturally identify an order $\sigma \in S_n$ (permutation on $\{x_1, \ldots, x_n\}$) in which the ROABP "reads the variables". Indeed, there are polynomials which are computable by poly(n)-width ROABPs

[1] A non-commutative polynomial is one in which the variables do not commute, i.e. $xy \neq yx$.

in one order, but require exponential width in a different order. In fact, a straight-forward application of Nisan's characterisation shows that the $2n$-variate polynomial $(x_1+y_1)(x_2+y_2)\cdots(x_n+y_n)$ is computable by a width-2 ROABP in the order $(x_1,y_1,x_2,y_2,\ldots,x_n,y_n)$; but any ROABP that reads all the **x**-variables before the **y**-variables (e.g. in the order $(x_1,\ldots,x_n,y_1,\ldots,y_n)$) requires width $2^{\Omega(n)}$. The existence of such polynomials naturally leads to the following classes of polynomials (defined in Section 2).

- ROABP$[\exists](n,d,w)$ - n-variate, individual degree d polynomials that are computable by a width-w ROABP in *some* order $\sigma \in S_n$.
- ROABP$[\forall](n,d,w)$ - n-variate, individual degree d polynomials that are computable by a width-w ROABP in *every* order.

Clearly, ROABP$[\forall](n,d,w) \subseteq$ ROABP$[\exists](n,d,w)$, and the former class requires exponential width to simulate the latter, due to the example discussed above.

Observe that an ROABP in the order id $=(x_1,\ldots,x_n)$, can be written as $\mathbf{u}^\intercal \cdot M_1(x_1)\cdot M_2(x_2)\cdots M_n(x_n)\cdot\mathbf{v}$, with entries of each M_i being univariate polynomials in $\mathbb{C}[x_i]$. Alternatively, we can view the same, as $\mathbf{u}^\intercal\left(\prod_{i\in[n]}\left(A_{i,0}+A_{i,1}x_i+\cdots+A_{i,d}x_i^d\right)\right)\mathbf{v}$, by interpreting each M_i as a univariate with matrices as coefficients. We refer to these matrices $\{A_{i,j}\}$ as the *coefficient matrices* of the ROABP.

Now based on the properties of the coefficient matrices $\{A_{i,j}\}$, one can define the following models and the corresponding classes.

- *Commutative ROABPs:* ROABPs where all the $n(d+1)$ coefficient matrices commute with each other (see Definition 12).
  $\mathsf{commROABP}(n,d,w)$ - n-variate, individual degree d polynomials that are computable by a width w commutative ROABP.
- *Diagonal ROABPs:* ROABPs where all the $n(d+1)$ coefficient matrices are diagonal matrices (see Definition 13).
  $\mathsf{diagROABP}(n,d,w)$ - n-variate, individual degree d polynomials that are computable by a width w diagonal ROABP.

First of all, $\mathsf{commROABP}(n,d,w) \subseteq$ ROABP$[\forall](n,d,w)$ for any n,d,w, since the coefficient matrices in any commutative ROABP are commutative, and one can multiply the matrices in any order to get the same result. Likewise, as all diagonal matrices commute with each other, $\mathsf{diagROABP}(n,d,w) \subseteq \mathsf{commROABP}(n,d,w)$. In this paper, we investigate commutative and diagonal ROABPs to understand if and when these two classes are the essentially (up to polynomial-factors) equal.

While it is indeed true that even diagonal ROABPs are universal, it is reasonable to ask if there are any interesting polynomial families that are efficiently computable by commutative and diagonal ROABPs. In this regard, let us begin by looking at the constructions of "all-order-ROABPs" for two well studied polynomial families: *elementary symmetric polynomials* and *powers of linear forms*. Incidentally, these constructions can naturally be interpreted as commutative ROABPs, and further, they even lead to diagonal ROABPs that achieve the best known upper bounds. We believe that these examples should serve as an additional motivation to study the models of commutative and diagonal ROABPs.

▶ **Definition 1** (Elementary Symmetric Polynomials). *The n-variate elementary symmetric polynomial of degree d, denoted by ESym_n^d is defined as follows.*

$$\mathsf{ESym}_n^d(\mathbf{x}) := \sum_{\substack{S\subset[n]\\|S|=d}}\prod_{i\in S} x_i \tag{1.1}$$

Following is a folklore construction (with a minor tweak) of an ROABP for ESym_n^d which is provably tight owing to the characterisation result by Nisan [14] (see full version). We illustrate the construction for $n = 5$ and $d = 3$ in the full version, and give the general recipe here without a proof of correctness.

▶ **Construction 1.2.** *For any $n, d \in \mathbb{N}$ such that $d \leq n$, we have the following.*

$$\mathsf{ESym}_n^d(\mathbf{x}) = (M(x_1)M(x_2)\cdots M(x_n))\,[1, d+1],$$

where for all i, $M(x_i)$ is a $(d+1) \times (d+1)$ matrix such that $M(x_i)[k,k] = 1$ for all $1 \leq k \leq (d+1)$, and $M(x_i)[k,k+1] = 1$ for all $1 \leq k \leq d$; all other entries of $M(x_i)$ are zero.

The matrix $M(x_i)$ can also be written as $(I + Ax_i)$, where A is a matrix with 1s on its super-diagonal and zeros everywhere else, and I is the identity matrix. This gives the expression: $\mathsf{ESym}_n^d(\mathbf{x}) = ((I + Ax_1)(I + Ax_2)\cdots(I + Ax_n))_{(1,d+1)} = \mathbf{u}^\intercal \left(\prod_{i\in[n]}(I + Ax_i)\right)\mathbf{v}$, for the obvious choice of $\mathbf{u}, \mathbf{v} \in \mathbb{C}^{(d+1)}$.

We can now make the following sequence of simple observations about this construction.

- All the coefficient matrices of the above ROABP: I and A, commute with each other. Thus, it is a commutative ROABP.
- $(I+Ax_1)(I+Ax_2)\cdots(I+Ax_n) = \sum_{0\leq j\leq n} \mathsf{ESym}_n^j\, A^j = \sum_{0\leq j\leq d} \mathsf{ESym}_n^j\, A^j$, since $A^j = 0$ for all $j \geq (d+1)$.
- For every $0 \leq j \leq d$, only the jth power of A that has a 1 in the $(1, 1+j)$th entry. Therefore, the $(1, d+1)$th entry of $(I + Ax_1)(I + Ax_2)\cdots(I + Ax_n)$ exactly computes the coefficient of A^d, which is ESym_n^d.

This perspective along with elementary interpolation, then leads us to the following *depth-3-multilinear* circuit for ESym_n^d of *top fan-in* $(n + 1)$ for all values of d, that is attributed to Ben-Or ([21]). This also happens to give the following nearly-optimal construction for a diagonal ROABP computing ESym_n^d.

▶ **Construction 1.3.** *For any $n, d \in \mathbb{N}$ and distinct $a_0, a_1, \ldots, a_n \in \mathbb{F}$, there exist constants $\beta_0, \beta_1, \ldots, \beta_n \in \mathbb{F}$ such that*

$$\mathsf{ESym}_n^d(\mathbf{x}) = \sum_{0\leq j\leq n} \beta_j(1 + a_jx_1)(1 + a_jx_2)\cdots(1 + a_jx_n)$$

Just as the commutative ROABP for $\mathsf{ESym}_n^d(\mathbf{x})$ leads us to Ben-or's construction of a diagonal ROABP, we also observe that the commutative ROABP computing dth power of an n-variate linear form $(x_1 + x_2 + \cdots + x_n)^d$ gives us the *duality trick* of Saxena [18] (see e.g. [19, Lemma 17.13]). We refer the interested reader to the full version.

As the coefficient matrices of diagonal ROABPs are diagonal matrices it is not difficult to observe that they are exactly *sums-of-products-of-univariates.* Thus, from the duality trick, we observe that diagonal ROABPs can efficiently simulate *diagonal depth* 3 *circuits* (a.k.a. *depth*-3 *powering circuits*) denoted by $\Sigma\wedge\Sigma$. That is, $\Sigma\wedge\Sigma(n, d, s) \subseteq \mathsf{diagROABP}(n, d, O(n, d, s))$.Also, a separation between these two classes is known due to the exponential lower bound from [15] for $x_1 \ldots x_n$ against the model $\Sigma \wedge \Sigma$. In essence, we have the following containments between classes[2], where each $\mathcal{C}$ stands for the class of n-variate, degree-d polynomials whose $\mathcal{C}$-size is poly(n, d).

$$\Sigma \wedge \Sigma \subsetneq \mathsf{diagROABP} \subseteq \mathsf{commROABP} \subseteq \mathrm{ROABP}[\forall] \subsetneq \mathrm{ROABP}[\exists]$$

[2] We have more intricate relationships between classes concerning ROABPs. See Subsection 1.3

Looking at the above hierarchy, we firstly realise that nearly optimal separations are known at the two “extremes”, but nothing is known about the intermediate levels. Further, since the intermediate levels are far more algebraically structured (coefficient matrices arising from special commutative algebras), it is reasonable to expect finer separations for these classes. Unfortunately, all the lower bounds that we know for diagonal and commutative ROABPs are those that are known for ROABP[$\forall$].

Secondly, even though diagonal ROABPs (*sum-of-products-of-univariates*) may be of independent interest as they subsume $\Sigma\wedge\Sigma$ circuits, they are also interesting from the point of view of polynomial identity testing. Owing to the algebraic structure of their coefficients, one can expect efficient PIT algorithms for these classes. But again, the best PIT algorithms that we know for diagonal and commutative ROABPs are those we know for ROABP[$\forall$]. We discuss more about polynomial identity testing algorithms for these classes in the full version.

1.1 Our Results

We now move to the central questions addressed in this article. In particular, we wish to understand if the classes commROABP and diagROABP are equal up to polynomial factors; this can be more formally stated as follows.

▶ **Question 1.4.** *Given an n-variate, individual degree d polynomial $f(\mathbf{x})$ computable by a width w commutative ROABP(i.e. $f \in \mathsf{commROABP}(n,d,w)$), does there exist a diagonal ROABP computing f of width $\mathrm{poly}(n,d,w)$?*

A measure that is often used to prove lower bounds against structured models (e.g. almost every lower bound against $\Sigma\wedge\Sigma$, and more recently [11]) is the *dimension of partial derivatives*, a complexity measure which was introduced by Nisan and Wigderson [15] (see Definition 16). For any polynomial $f \in \mathbb{C}[\mathbf{x}]$, the partial derivative complexity of f (denoted by $\mathrm{DPD}(f)$) is the dimension of the space spanned by *all* the partial derivatives of f. Nisan and Wigderson [15] observed that any n-variate, degree d polynomial $f(\mathbf{x})$ that has a $\Sigma\wedge\Sigma$ circuit of size s has $\mathrm{DPD}(f) \leq s(d+1)$. Therefore it is natural to ask whether the $\Sigma\wedge\Sigma$-size of every polynomial f is polynomially related to its dimension of partial derivatives. We formalize this question as follows.

▶ **Question 1.5.** *Does there exist a* constant c *such that for any n-variate, degree-d polynomial $f(\mathbf{x})$ with $\mathrm{DPD}(f) \leq s$, we have that the smallest $\Sigma\wedge\Sigma$ circuit that computes $f(\mathbf{x})$ has size at most $(nds)^c$?*

The size of the smallest $\Sigma\wedge\Sigma$ circuit for a polynomial is a well studied notion called the *Waring rank of f* (denoted by $\mathrm{WR}(f)$). Question 1.5 essentially asks if the Waring rank and the dimension partial derivatives of a polynomial are same up to polynomial factors. Unfortunately, at the moment we do not know the answers to either Question 1.4 or Question 1.5. However, our main result gives a rather surprising connection between Question 1.4 and Question 1.5. Specifically, we show that an positive answer to Question 1.5 answers Question 1.4 in the affirmative!

▶ **Theorem 2.** *For any $n, r \in \mathbb{N}$, let $S(r,m)$ denote the smallest $\Sigma\wedge\Sigma$-size required to compute any r-variate polynomial f with $\mathrm{DPD}(f) \leq m$.*
Then for all $n, d, w \in \mathbb{N}$, $\mathsf{commROABP}(n,d,w) \subseteq \mathsf{diagROABP}\left(n, d, S(w^2, w^2)nw^4\right)$.

▶ Remark 3. In fact, it can be inferred from our proof that a super-polynomial separation between commROABP and diagROABP will yield an explicit polynomial that witnesses a super-polynomial separation between dimension of partial derivatives and Waring rank. We elaborate on this in Remark 22.

A different (and perhaps equally surprising) consequence of Theorem 2 is that a super-polynomial separation between commutative ROABPs and diagonal ROABP will also give a super-polynomial separation dimension of partial derivatives and Waring rank. Note that not only do we not know the answers to Question 1.4 or Question 1.5, it is somewhat frustrating that we do not even know of a candidate polynomial that could potentially separate these classes. We expect that our analysis of these models that goes into proving the result above could help in making some progress in either of these questions.

1.2 An overview of the proof

We start by asking when diagonal ROABPs can efficiently simulate commutative ROABPs. This question naturally leads us to study properties of matrices that commute with each other. In particular, we analyse *commutative rings* generated by matrices that commute with each other.

A very high level overview. The results of Marinari, Möller, Mora [12], and Möller and Stetter [13] provide a characterisation of commutative rings of $w \times w$ matrices in terms of polynomials whose *dimension of partial derivatives* is at most $\mathrm{poly}(w)$. In the special case when these matrices are all diagonal, the same polynomials happen to have *Waring rank* at most w. Further, we observe that if the polynomials corresponding to a n-variate, width-w commutative ROABP have *Waring rank* at most s, then it can be simulated by a diagonal ROABP of width $\mathrm{poly}(n, w, s)$. This is essentially our main result. We now explain the characterisation given by [12] and [13] in a bit more detail.

Characterising rings of matrices

Consider the ring generated by a $w \times w$ matrix A, given by $\mathbb{C}[A] := \{q(A) : q(t) \in \mathbb{C}[t]\}$. The ring has at most w *linearly independent* matrices, as the characteristic polynomial of A gives a way to express A^w as a linear combination of lower powers of A. In fact, the ring $\mathbb{C}[A]$ is characterised by the *ideal* of all polynomials that are divisible by the *minimal polynomial of A* (see Fact A.1). This characterisation has an appropriate analogue for general matrix rings, as follows.

Suppose that $A_1, \ldots, A_r \in \mathbb{C}^{w \times w}$ commute with each other, and let $\mathbb{C}[A_1, \ldots, A_r]$, defined as $\{g(A_1, \ldots, A_r) : g(\mathbf{t}) \in \mathbb{C}[\mathbf{t}]\}$, be the ring generated by them[3]. Analogous to the univariate (singly-generated) case, we then consider the *ideal of dependencies* for the matrices $A_1, \ldots, A_r$: $J = \{p(\mathbf{t}) \in \mathbb{C}[\mathbf{t}] : p(A_1, \ldots, A_r) = 0\}$. As it turns out, $\mathbb{C}[A_1, \ldots, A_r]$ is indeed characterised by the ideal J (see Lemma 23).

Before delving further into the ideal of dependencies, we remark a structural property of polynomials that admit a diagonal ROABP of a certain width.

Understanding diagonal ROABPs. Consider the diagonal ROABP (depth-3 multilinear circuit) for the *elementary symmetric polynomial* $\mathsf{ESym}_{n,d}$ that is attributed to Ben-Or (see e.g. [21]). One first constructs the polynomial $g(t, \mathbf{x}) := (1 + tx_1)(1 + tx_2) \cdots (1 + tx_n)$, and then obtains $\mathsf{ESym}_{n,d}$ as the coefficient of t^d in $g(t, \mathbf{x})$, using interpolation. It turns out that any diagonal ROABP computing a polynomial $f(\mathbf{x})$ can similarly be seen as expressing f as a linear combination of evaluations of a *low-degree* $g(t, \mathbf{x})$ that is a "product of univariates"

[3] Any ring of $w \times w$ matrices is generated by at most w^2 matrices.

(see Observation 18). Here, the number of evaluations needed is *equal* to the width of the ROABP. Moreover the converse of this statement is also true, thus giving us an equivalent formulation for diagonal ROABPs.

Therefore, we analyse the ideal J with the goal of expressing the corresponding commutative ROABP as a *sum of* $\mathbf{t}$*-evaluations* of some $G(\mathbf{t}, \mathbf{x}) = G_1(\mathbf{t}, x_1) \cdot G_2(\mathbf{t}, x_2) \cdots G_n(\mathbf{t}, x_n)$.

The ideal of dependencies. Let us first make our statement about $\mathbb{C}[A_1, \ldots, A_r]$ being characterised by J a bit more precise: there is a *ring-isomorphism* between $\mathbb{C}[A_1, \ldots, A_r]$ and the *quotient ring* ${}^{\mathbb{C}[\mathbf{t}]}/_{J}$. Therefore it is crucial to understand J (and ${}^{\mathbb{C}[\mathbf{t}]}/_{J}$) to understand the ring of matrices, in order to move towards the above mentioned goal.

Let $p(t)$ be the minimal polynomial of some matrix A, and consider the ideal $\langle p \rangle$. If $p(t) = (t-5)^3$, then we know that any $q(t)$ belongs to $\langle p \rangle$ *if and only if* the first 3 derivatives of $q(t)$ vanish at $t = 5$; i.e. $q(5) = q'(5) = q''(5) = 0$. In general, for $p(t) = (t-a_1)^{e_1}(t-a_2)^{e_2} \cdots (t-a_k)^{e_k}$, membership in the ideal $\langle p \rangle$ is *characterised* by the first e_i derivatives vanishing at $t = a_i$, for *each* $i = 1, 2, \ldots, k$. Moreover, the polynomial "$q(t) \bmod p(t)$" can be obtained by applying a *linear transformation* on the evaluations of the $e_1, \ldots, e_k$ derivatives at the respective points $a_1, \ldots, a_k$.

We now extend this understanding to the multivariate setting. We already have the correct analogue for $\langle p \rangle$, which we call the ideal of dependencies J. Next, we need a characterisation for "$g(\mathbf{t}) \bmod J$" in terms of some derivatives of $g(\mathbf{t})$ evaluated at some points related to J. While these choices were quite clear in the univariate setting from p; the multivariate setting requires a little more care. Fortunately for us, the works of Marinari, Möller, Mora [12], and Möller and Stetter [13] provide an adequate solution.

Firstly, observe that J has a finite *variety* (common zeroes of all polynomials in J). Thus the variety $\mathbf{V}(J)$ is a good multivariate analogue for the set of evaluation points. The other ingredient that we require is a compatible notion of "multiplicity of J" at a point $\bar{\alpha}$ in its variety. For this, [12] look at the set of all *partial derivative operators* (see Definition 25) which map *every* polynomial in J to a polynomial that vanishes at $\bar{\alpha}$. These operators form a vector space over $\mathbb{C}$, and the "multiplicity of J at $\bar{\alpha}$" is then defined as the *dimension* of this vector space.

In the univariate setting, the multiplicity of q at a point a_i is defined as the *highest* number e_i such that the first e_i derivatives of q vanish at the point a_i. Thus, one can naturally identify a "highest derivative", with the other derivatives being its "down-shifted versions". Analogously, the derivative operator space corresponding to J and a point $\mathbf{v} \in \mathbf{V}(J)$ is *closed under taking down-shifts* (see Definition 29). An ideal J with $\mathbf{V}(J) = \{\bar{\alpha}_1, \ldots, \bar{\alpha}_k\}$, is then captured by a collection of z vector spaces of derivative operators $\Delta_1, \Delta_2, \ldots, \Delta_k$, in the following sense (see Lemma 32).

- For each $i \in [k]$, Δ_i corresponds to the point $\bar{\alpha}_i$ and is down-closed.
- Dimension of the quotient ring ${}^{\mathbb{C}[\mathbf{t}]}/_{J}$ is $w = \dim(\Delta_1) + \dim(\Delta_2) + \cdots + \dim(\Delta_k)$.
- Let $\{D_{i,1}, \ldots, D_{i,w_i}\}$ be a basis of Δ_i. Then there exists a map $\Phi : \mathbb{C}^w \to {}^{\mathbb{C}[\mathbf{t}]}/_{J}$ such that for any polynomial $q(\mathbf{t})$, Φ maps the w values: $\{D_{i,j}(q)(\mathbf{v}_i)\}$, to the "remainder polynomial" $(q(\mathbf{t}) \bmod I)$.

Further, Möller and Stetter [13] show that the map Φ stated above is just a linear transformation (see Lemma 36).

Consequences for ROABPs. We now outline the proof of our main result (Theorem 2).

- Given a commutative ROABP $f(\mathbf{x}) = \mathbf{b}^{\mathsf{T}} \cdot \prod_{i \in [n]} \left(A_{i,0} + A_{i,1}x_i + \cdots + A_{i,d}x_i^d\right) \cdot \mathbf{c}$ of width w, we define $F(\mathbf{x}) := \prod_{i \in [n]} \left(A_{i,0} + A_{i,1}x_i + \cdots + A_{i,d}x_i^d\right)$ to be a matrix of polynomials. Then, $f(\mathbf{x})$ is just a linear combination (given by $\mathbf{b}\mathbf{c}^{\mathsf{T}}$) of the entries of F.

- We then identify a set of matrices $A_1, \ldots, A_r$ that generate the coefficient-matrix-ring; i.e. $\mathbb{C}[A_1, \ldots, A_r] = \mathbb{C}[A_{1,0}, \ldots, A_{1,d}, \ldots, A_{n,d}]$. As we can always use the coefficient matrices themselves, and because we are dealing with $w \times w$ matrices, $r \leq \min(w^2, n(d+1))$.
- Let J be the ideal of dependencies for $A_1, \ldots, A_r$ and suppose the normal set of J (see Definition 35) has size, say $m \leq w^2$. Then each $A_{i,j}$ is a polynomial in $A_1, \ldots, A_r$ that has $\leq m$ monomials.
- For each i, j, suppose $G_{i,j}(t_1, \ldots, t_r)$ is the polynomial such that $G_{i,j}(A_1, \ldots, A_r) = A_{i,j}$; the entries of $A_{i,j}$ are linear combinations of $\mathbf{t}$-coefficients of $G_{i,j} = (G_{i,j} \bmod J)$. Then we observe that $G(\mathbf{t}, \mathbf{x}) := \prod_{i \in [n]} \left(G_{i,0}(\mathbf{t}) + G_{i,1}(\mathbf{t})x_i + \cdots + G_{i,d}(\mathbf{t})x_i^d\right)$, such that $G(A_1, \ldots, A_r, \mathbf{x}) = F(\mathbf{x})$. This means that even $f(\mathbf{x})$ is a linear combination of the $\mathbf{t}$-coefficients of $(G(\mathbf{t}, \mathbf{x}) \bmod J)$, as it is a linear combination of the entries of $F(\mathbf{x})$. We prove this in Lemma 20.
- Now let $\mathbf{V}(J) = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ and for each $\ell \in [k]$ let $\{D_{\ell,1}, \ldots, D_{\ell,m_\ell}\}$ be a basis for the derivative operator space corresponding to $\mathbf{v}_\ell$. Then from the results of [12, 13] we get that for any $g(\mathbf{t})$, every $\mathbf{t}$-coefficient of $(g(\mathbf{t}) \bmod J)$ is a fixed linear combination of the m values given by $(D_{\ell,*}(g))(v_\ell)$.
- This brings us one step away from our goal of expressing $f(\mathbf{x})$ as a linear combination of $\mathbf{t}$-evaluations of some $G(\mathbf{t}, \mathbf{x})$ which is a product of univariates. What we need is a way to express each of $(D_{\ell,*}(G))(\mathbf{v})$ as a linear combination of $\mathbf{t}$-evaluations of $G(\mathbf{t}, \mathbf{x})$.
- It turns out that the number of evaluations of $G(\mathbf{t}, \mathbf{x})$ required to compute $(D_{\ell,*}(G))(\mathbf{v})$ is $\mathrm{poly}(\deg(h_{\ell,*}), \mathrm{WR}(h_{\ell,*}))$, where $h_{\ell,*}$ is the *polynomial corresponding to* $D_{\ell,*}$ (see paragraph below Definition 25). This is a non-trivial fact; we prove it in Lemma 21.
- Finally, since each space Δ_ℓ is *down-closed*, we have that the dimension of partial derivatives $\mathrm{DPD}(h_{\ell,*}) \leq \dim(\Delta_\ell) \leq m$ for each $h_{\ell,*}$. Therefore, using the hypothesis that $\mathrm{WR}(h) = \mathrm{poly}(r, \mathrm{DPD}(h))$ for any r-variate h, we get that $(D_{\ell,*}(G))(\mathbf{v})$ can be expressed as a linear combination of $\mathrm{poly}(r, \mathrm{DPD}(h_{\ell,*}), \deg(h_{\ell,*})) = \mathrm{poly}(r, m)$ evaluations of $G(\mathbf{t}, \mathbf{x})$ for each $D_{\ell,*}$.
- Combining all the above observations, we can see that the hypothesis implies that $f(\mathbf{x})$ can indeed be written as a linear combination of $\mathrm{poly}(r, m) = \mathrm{poly}(n, d, w)$ evaluations of $G(\mathbf{t}, \mathbf{x})$, thereby proving Theorem 2.

1.3 Landscape of ROABP classes

As mentioned earlier, although Theorem 2 relates Question 1.4 and Question 1.5, the answer to both these questions remain unknown. In this regard, we would like to conjecture that the answer to both questions is false.

▶ **Conjecture 4.** *There exists an explicit n-variate degree d polynomial $f(\mathbf{x})$ such that $f \in \mathsf{commROABP}(n, d, \mathrm{poly}(n, d))$ and any diagonal ROABP computing f requires width $n^{\omega(1)}$.*

▶ **Conjecture 5.** *There exists an explicit n-variate polynomial $f(\mathbf{x})$ of degree* $\mathrm{poly}(n)$ *such that* $\mathrm{DPD}(f) = \mathrm{poly}(n)$ *but* $\mathrm{WR}(f) = n^{\omega(1)}$.

Even though many would agree that Conjecture 4 and Conjecture 5 are probably true, we do not even know of any candidate polynomial that will witness the truth of this conjecture. In relation to this, we remark that the following statement can be inferred from our proof of Theorem 2. If there exists a commutative ROABP of width $\mathrm{poly}(n, d)$ computing an n-variate, degree-d polynomial f, which requires diagonal ROABPs of super-polynomial width, then the commutative ROABP for f gives a different explicit polynomial h that has polynomial dimension of partial derivatives, but has super-polynomial Waring rank (see Remark 22 for details). As a result, even a candidate polynomial for proving Conjecture 4 remains unknown.

In the context of Conjecture 4, we remark the following connection between diagonal ROABPs and tensor rank.

▶ Remark 6. Observe that the width of a diagonal ROABP exactly captures the *tensor rank* of the corresponding *tensor*. A tensor $T : [d]^n \to \mathbb{C}$ of order[4] n can naturally be viewed as a polynomial $f_T = \sum_{\mathbf{i}\in[d]^n} T(i_1, \ldots, i_n)x_1^{i_1} \cdots x_n^{i_n}$. The (tensor) rank of any T (denoted by $\mathsf{TR}(T)$) is the smallest r such that T can be expressed as sum of r elementary tensors. Thus for any tensor T, $\mathsf{TR}(T) = r$ if and only if $f_T(\mathbf{x})$ can be expressed as sum of r many products of univariates; which immediately implies $\mathsf{diagROABP}(n, d, w) = \{f_T \in \mathbb{C}[\mathbf{x}] \mid \mathsf{TR}(T) \le w\}$. Obtaining strong lower bounds on the rank of explicit tensors is a major open problem in algebraic complexity theory, where the goal is to obtain an explicit tensor T of order-n such that $\mathsf{TR}(T) = d^{n(1-o(1))}$ (see e.g. [17]).

Remark 6 tells us that proving strong width lower bounds against diagonal ROABPs could potentially imply lower bounds on the rank of explicit tensors. While this could partially explain why there are no separations between diagonal ROABPs and commutative or “all-order” ROABPs, it is also worth mentioning that order-n tensors for a *growing parameter* n are rarely studied in the context of tensor rank lower bounds.

With regard to Conjecture 5, we briefly discuss some known results about the problem of computing the dimension of the partial derivative space.

Shitov [20] showed that given any degree 3 polynomial f in its sparse representation, computing $\mathrm{WR}(f)$ is NP-hard, by reducing it to computing the tensor rank of order 3 *symmetric tensors*. On the other hand, when a polynomial f is presented in its sparse representation (as sum of monomials), García-Marco, Koiran, Pecatte and Thomassé [7] prove that computing the dimension of the partial derivative space is $\#\mathsf{P}$-hard (not known to be $\#\mathsf{P}$-complete). Thus, even though computing Waring rank is a hard problem, it is not quite clear if disproving Conjecture 5 goes against it. Moreover, it is possible that Waring rank is easy to *approximate* up to polynomial factors, which is all that a disproof of Conjecture 5 would imply. On a related note, Kayal [9] gave a randomised $\mathrm{poly}(n, d)$-time algorithm to compute the waring rank of an n-variate, degree-d polynomial that is given as a blackbox (in the *non-degenerate case*).

Although the results in this article entirely concern Question 1.5 and Question 1.4, there are several other interesting open questions surrounding the landscape of complexity classes involving ROABPs. We discuss these interconnections between ROABP classes now, and later illustrate them in Figure 1.

Let us consider the class of polynomials computed by ROABPs that remain unchanged by interchanging layers in the branching program[5]. We prefer to use the term *layer-commutative ROABPs* (denoted by $\mathsf{layer\text{-}commROABP}(n, d, w)$) to denote the class of n-variate degree d polynomials computed by an ROABPs such that if $f = u^T M_1(x_1) \cdots M_n(x_n)v$ then the matrices of univariate polynomials $M_1, \ldots, M_n$ commute. That is, $M_i(x_i)M_j(x_j) = M_j(x_j)M_i(x_i)$ for all $i, j \in [n]$. Clearly, $\mathsf{layer\text{-}commROABP}(n, d, w) \subseteq \mathrm{ROABP}[\forall](n, d, w)$, and $\mathsf{commROABP}(n, d, w) \subseteq \mathsf{layer\text{-}commROABP}(n, d, w)$. This immediately leads us to the following two open questions whose answer seems unclear at the moment.

[4] Commonly used term in the literature about tensors; not be confused with the order of an ROABP.

[5] The class $\mathrm{ROABP}[\forall](n, d, w)$ has been studied in the context of PIT, and is sometimes called *commutative ROABPs* in some works (e.g. [8]). We use a different notation to avoid any ambiguity.

▶ **Question 1.6.**

1. *Are* $\mathsf{layer\text{-}commROABP}(n,d,w)$ *and* $\mathrm{ROABP}[\forall](n,d,w)$ *equivalent up to a polynomial blow-up in the width* w*?*
2. *Are* $\mathsf{commROABP}(n,d,w)$ *and* $\mathsf{layer\text{-}commROABP}(n,d,w)$ *equivalent up to a polynomial blow-up in the width* w*?*

We hope that a better understanding the algebra associated with commutative ROABPs may shed light on the answers to above questions.

Along with the complexity of computing polynomials exactly, another notion that is considered in algebraic complexity theory and more specifically in *geometric complexity theory*, is *border complexity* of polynomials. Let $\mathcal{C}$ be a class of polynomials. We say that f is in the class $\overline{\mathcal{C}}$ (border of $\mathcal{C}$), if f can be "arbitrarily-approximated" by a circuit in $\mathcal{C}$. That is, there exists a polynomial $g(\epsilon) \in \mathbb{C}(\epsilon)$ in class $\mathcal{C}$ such that $f = \lim_{\epsilon \to 0} g$. The *border-complexity* of f is then at most the size of the circuit computing g. Clearly, $\mathcal{C} \subseteq \overline{\mathcal{C}}$. Understanding whether $\mathcal{C} = \overline{\mathcal{C}}$ for interesting classes such as VP and VBP are major open problems in algebraic complexity theory. Here, we are interested in the case when $\mathcal{C} = \mathsf{diagROABP}(n,d,w)$ (defined in Definition 17).

▶ **Question 1.7.** *Is there a super-polynomial separation between the classes* $\mathsf{diagROABP}(n,d,w)$ *and* $\overline{\mathsf{diagROABP}}(n,d,w)$*?*

As $\mathsf{diagROABP}(n,d,w) = \{f_T \in \mathbb{C}[\mathbf{x}] \mid \mathsf{TR}(T) \leq w\}$, we have $\overline{\mathsf{diagROABP}(n,d,w)} = \{f_T \in \mathbb{C}[\mathbf{x}] \mid \underline{\mathsf{TR}}_{\mathbb{C}}(f) \leq w\}$. Here, $\underline{\mathsf{TR}}(f)$ denotes the border rank of tensors. Border rank of tensors is studied extensively in several contexts for instance, border rank of *matrix multiplication tensor* is used to obtain bounds on the arithmetic complexity of matrix multiplication. In this setting, the order of the tensor is usually bounded by a constant, and this setting slightly deviates from the main theme algebraic circuit complexity.

It can be checked that just like $\mathsf{commROABP}(n,d,w)$, $\overline{\mathsf{diagROABP}}(n,d,w)$ is also contained in $\mathrm{ROABP}[\forall](n,d,w)$ (because ROABP-complexity is characterised by rank, which is a continuous measure). However, it is unclear if these two ways of "generalising" diagonal ROABPs have different computational powers. This brings us to the following question.

▶ **Question 1.8.** *Are the classes* $\overline{\mathsf{diagROABP}}(n,d,w)$ *and* $\mathsf{commROABP}(n,d,w)$ *equivalent up to polynomial factors?*

Note that Question 1.8 is linked to the question of understanding $\mathsf{commROABP}(n,d,w)$ and $\mathrm{ROABP}[\forall](n,d,w)$ in Question 1.6. Also, answering this question in the affirmative is similar in spirit to the recent "de-bordering" results due to Dutta et al. [5]. They proved that the border of constant *top fan-in* depth three circuits is contained in the class VBP. Here, Question 1.8 is essentially asking if for the class of diagonal ROABPs (albeit with unbounded fan-in), the border is contained in a much simpler class of commutative ROABPs? However, answering this in the negative could potentially be as hard as (or even harder than) separating commutative ROABPs from diagonal ROABPs. In fact, it is not even clear if these two classes should be comparable (contained in one another). We believe that any answer to Question 1.8 would be an interesting development in algebraic complexity theory.

We summarize all the models and the interconnections between the structured ROABP classes in Figure 1.

2 Preliminaries

We now formally define the classes of polynomials and other algebraic models of computation that we study in this paper. First, we fix some notation.

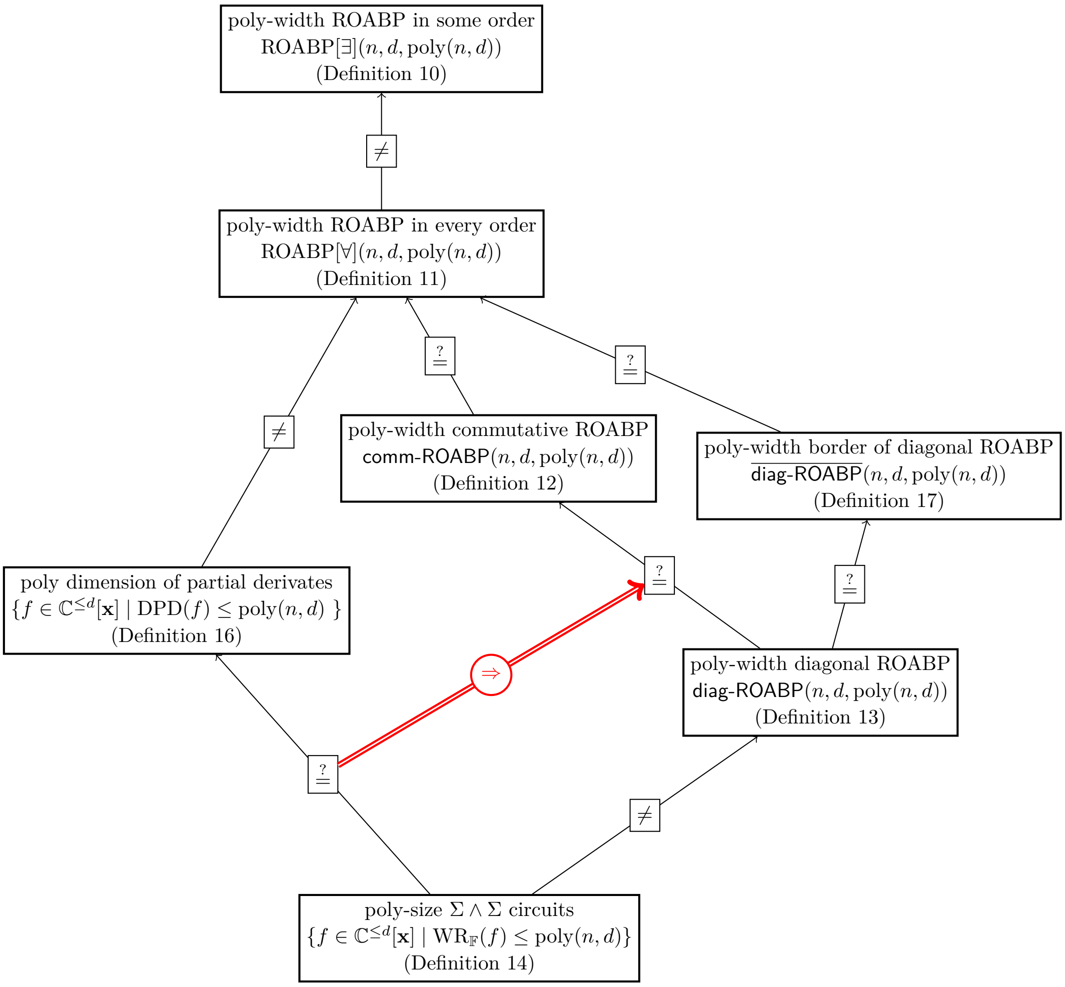

poly-width ROABP in some order
$\mathrm{ROABP}[\exists](n, d, \mathrm{poly}(n, d))$
(Definition 10)
$\neq$
poly-width ROABP in every order
$\mathrm{ROABP}[\forall](n, d, \mathrm{poly}(n, d))$
(Definition 11)
$\stackrel{?}{=}$
$\stackrel{?}{=}$
$\neq$
poly-width commutative ROABP
$\mathsf{comm\text{-}ROABP}(n, d, \mathrm{poly}(n, d))$
(Definition 12)
poly-width border of diagonal ROABP
$\overline{\mathsf{diag\text{-}ROABP}}(n, d, \mathrm{poly}(n, d))$
(Definition 17)
poly dimension of partial derivates
$\{f \in \mathbb{C}^{\leq d}[\mathbf{x}] \mid \mathrm{DPD}(f) \leq \mathrm{poly}(n, d)\ \}$
(Definition 16)
$\stackrel{?}{=}$
$\stackrel{?}{=}$
$\Rightarrow$
poly-width diagonal ROABP
$\mathsf{diag\text{-}ROABP}(n, d, \mathrm{poly}(n, d))$
(Definition 13)
$\stackrel{?}{=}$
$\neq$
poly-size $\Sigma \wedge \Sigma$ circuits
$\{f \in \mathbb{C}^{\leq d}[\mathbf{x}] \mid \mathrm{WR}_{\mathbb{F}}(f) \leq \mathrm{poly}(n, d)\}$
(Definition 14)

Figure 1 The ROABP landscape: edges denote bottom-up inclusion, Theorem 2 is in red.

Notation

- We use the shorthand $[n]$ to denote the set $\{1, 2, \ldots, n\}$.
- We use boldface letters like $\mathbf{x}, \mathbf{t}, \mathbf{A}, \mathbf{b}$, to denote sets/vectors. The individual elements/-coordinates are denoted by indexed versions of the same characters: $\mathbf{A} = \{A_1, \ldots, A_r\}$. Whenever the size of these sets is not clear from context, we denote them using subscripts: $\mathbf{x}_{[n]} = \{x_1, \ldots, x_n\}$.
- For a polynomial $f(\mathbf{x})$, we denote *support of* f the set of monomials appearing in f with a nonzero coefficient by $\mathrm{supp}(f)$.
- For $\mathbf{x} = \{x_1, \ldots, x_n\}$, and any vector $\mathbf{e} \in \mathbb{N}^n$, we use the shorthand $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$.
- For a polynomial $f(\mathbf{x})$ and a monomial $\mathbf{x}^{\mathbf{e}}$, we use $\partial_{\mathbf{e}} f$ to denote the partial derivative $\frac{\partial^{|\mathbf{e}|} f}{\partial x_1^{e_1} \cdots \partial x_n^{e_n}}$.
- For a matrix M, $M[i, j]$ denotes its (i, j)th entry.

We start by defining *algebraic circuits* and *algebraic branching programs*. Note that we work with the field of complex numbers unless mentioned otherwise.

▶ **Definition 7** (Algebraic circuits). *An* algebraic circuit *is specified by a directed acyclic graph, with leaves (in-degree zero; also called* inputs*) labelled by field constants or variables, and internal nodes labelled by $+$ or $\times$. The nodes with out-degree zero are called the* outputs *of the circuit. Computation proceeds in the natural way, where inductively each $+$ gate computes the sum of its children and each $\times$ gate computes the product of its children.*

The size *of the circuit is defined as the number of nodes in the underlying graph.*

▶ **Definition 8** (Algebraic Branching Programs). *An algebraic branching program is a layered, directed graph. There are two special vertices, source s and sink t which are the only vertices in the first and last layers, respectively. All the edges in the graph are from one layer to the consecutive layer. Each edge is labelled by a univariate polynomial in the underlying variables over the underlying field. Each path from s to t computes the product of the edge labels and the ABP computes the sum of all the paths from s to t. Then, any ABP can be viewed as a product of matrices (each matrix having univariate polynomials as its entries) and the ABP computes the $(1,1)$th entry of the matrix product. The maximum number of vertices in a single layer (dimension of the largest matrix in the product) is called its* width. *The size of the ABP is the total number of vertices in it.*

We now define the various structured ROABPs and other related classes that are the main objects of interest in our paper. We start by defining the basic model of ROABPs.

▶ **Definition 9** (Read-once Oblivious ABPs). *Over the field $\mathbb{C}$ of complex numbers, a* read-once oblivious algebraic branching program *or an* ROABP*, computes an n-variate,* individual *degree d polynomial using a matrix-vector product of the following form.*

$$R(\mathbf{x}) = \mathbf{u}^{\mathsf{T}} \cdot M_1(x_{\sigma(1)}) \cdot M_2(x_{\sigma(2)}) \cdots M_n(x_{\sigma(n)}) \cdot \mathbf{v}$$

where

- *For each $i \in [n]$, the matrix $M_i(x_{\sigma(i)})$ has entries that are univariates of degree $\leq d$ in the variable $x_{\sigma(i)}$,*
- $\mathbf{u} \in \mathbb{C}^{w_0}$*, $M_1(x_{\sigma(1)}) \in (\mathbb{C}[x_{\sigma(1)}])^{w_0 \times w_1}$, $\ldots$, $M_i(x_{\sigma(i)}) \in (\mathbb{C}[x_{\sigma(i)}])^{w_i \times w_{i+1}}$, $\ldots$,* $\mathbf{v} \in \mathbb{C}^{w_n}$*,*
- *the* width *w of the ROABP R is defined as $w = \max\{w_0, w_1, \ldots, w_n\}$,*
- *the permutation σ is called as* the order of the ROABP R.

The following two subclasses of polynomials then follow naturally from the definition of ROABPs.

▶ **Definition 10** (ROABPs in some order). *For $n, d, w \in \mathbb{N}$, an n-variate polynomial $f(\mathbf{x})$ of* individual *degree d is said to have an ROABP of width w* in the order *$\sigma \in S_n$, if there exists a width w ROABP $R(\mathbf{x})$ that computes $f(\mathbf{x})$ in the order σ. We denote the class of such polynomials by* $\mathrm{ROABP}[\sigma](n,d,w)$*.*
Further, we use $\mathrm{ROABP}[\exists](n,d,w)$ *to denote the class of polynomials that have a width w ROABP in* some *order. That is,* $\mathrm{ROABP}[\exists](n,d,w) = \bigcup_{\sigma \in S_n} \mathrm{ROABP}[\sigma](n,d,w)$*.*

We can then extend this definition naturally as follows.

▶ **Definition 11** (ROABPs in every order). *For $n, d, w \in \mathbb{N}$, an n-variate polynomial $f(\mathbf{x})$ of* individual *degree d is said to have an ROABP of width w* in every order*, if for all permutations $\sigma \in S_n$, there exists a width w ROABP $R_{(\sigma)}(\mathbf{x})$ that computes $f(\mathbf{x})$ in the order σ.*
We denote this class of polynomials by $\mathrm{ROABP}[\forall](n,d,w)$*.*

Now, based on the properties of the *coefficient matrices*, we define the two subclasses of ROABPs that Theorem 2 talks about.

▶ **Definition 12** (Commutative ROABPs). *An n-variate, individual degree d ROABP of width w is called a* commutative ROABP *if its coefficient matrices are all $w \times w$ matrices that are (pairwise) commutative.*
We refer of the class of polynomials computed by such ROABPs by $\mathsf{commROABP}(n, d, w)$.

▶ **Definition 13** (Diagonal ROABPs). *An n-variate, individual degree d ROABP of width w is called a* diagonal ROABP *if its coefficient matrices are $w \times w$* diagonal *matrices. We refer of the class of polynomials computed by such ROABPs by* $\mathsf{diagROABP}(n, d, w)$.

Further, we define other concepts about polynomials like *depth-3 powering circuits*, *Waring rank* and *Tensor rank*, since we talk about the connections between them and subclasses of ROABPs defined above.

▶ **Definition 14** (Depth 3 powering circuits ($\Sigma \wedge \Sigma$)). *Over the field $\mathbb{C}$, a depth 3 powering circuit of size s, computes an n-variate, (total) degree d polynomial as an $\mathbb{C}$-linear combination of s terms, each of which is a $\leq d$th power of an $\mathbb{C}$-linear form in the underlying variables $x_1, \ldots, x_n$.*
That is, vectors $\mathbf{a}_1, \ldots, \mathbf{a}_s \in \mathbb{C}^{n+1}$, constants $\beta_1, \ldots, \beta_s$, and $d_1, d_2, \ldots, d_s \in \{0, \ldots, d\}$, define the following n-variate, degree-d, size s depth 3 powering circuit.

$$C(\mathbf{x}) = \sum_{i \in [s]} \beta_i \left(a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_n x_n\right)^{d_i}$$

▶ **Definition 15** (Waring rank). *For an n-variate, degree-d polynomial $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$, the* Waring rank *of f is defined to be the size of the smallest depth 3 powering circuit that computes it. We will denote the Waring rank of a polynomial f by* $\mathrm{WR}(f)$.

▶ **Definition 16** (Dimension of partial derivatives). *For an n-variate polynomial $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$, the* dimension of partial derivatives, *which we shall denote by* $\mathrm{DPD}(f)$ *is defined as* $\mathrm{DPD}(f) = \dim\left(\mathrm{span}_{\mathbb{C}}\left\{\partial_{\mathbf{e}} f : \mathbf{e} \in \mathbb{N}^n\right\}\right)$. *Here, $\partial_{\mathbf{e}} f$ denotes the partial derivative $\frac{\partial^{|\mathbf{e}|} f}{\partial x_1^{e_1} \cdots \partial x_n^{e_n}}$.*

Finally, we define the border of diagonal ROABPs as follows, which coincides with the definition of commonly known definition of *border-tensor-rank*.

▶ **Definition 17** (Border of diagonal ROABPs). *For any polynomial $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$, $f(\mathbf{x})$ is in the class $\overline{\mathsf{diagROABP}(n, d, w)}$ if there exists a polynomial $g \in \mathbb{C}(\epsilon)$ in the class $\mathsf{diagROABP}(n, d, w)$ such that $f = \lim_{\epsilon \to 0} g$.*

Organization of the paper

The proof the main theorem of this paper(Theorem 2) can be found in Section 4. The appendix is dedicated to studying the algebraic structure of commutative ROABPs, which gives us the necessary ingredients to prove the main theorem. There we first study the "singly-generated" case in Subsection A.1, followed by the structure of general commutative matrix rings in Subsection A.2.

3 Open questions

Owing to the connections of subclasses of ROABPs with other well-studied models, we believe that resolving any of the questions stated in Section 1 in any direction would be very interesting to the algebraic complexity community, and might even lead to new approaches for PIT of ROABPs and depth 3 powering circuits.

A specific follow-up question to our main theorem(Theorem 2) is that of finding an appropriate converse. For example, is it true that if diagonal ROABPs can efficiently simulate commutative ROABPs, then dimension of partial derivatives essentially captures the Waring rank of any polynomial? It is not clear how one would go about proving the above statement directly. For proving the contrapositive, the main technical challenge seems to be to arrive at a candidate commutative ROABP using a polynomial that would witness the separation between dimension of partial derivatives and Waring rank.

4 Proof of the main theorem

We start with an observation about diagonal ROABPs that gives an *equivalent* alternate view of the model, which will be useful for our results.

▶ **Observation 18** (Alternate view of diagonal ROABPs). *If $f(x_1,\ldots,x_n)$ has a* diagonal ROABP *of width w, then there is a polynomial $g(t,\mathbf{x})$ with $\deg_t(g) \leq nw$, such that $f(\mathbf{x}) = \sum_{j\in[w]} g(j,\mathbf{x})$.*

Proof. Suppose $f(\mathbf{x}) = \sum_{j\in[w]} \prod_{i\in[n]} f_{j,i}(x_i)$. Then we define polynomials $L_1(t),\ldots,L_w(t)$ such that for each $j,k \in [w]$, $L_j(k) = 1$ if $j = k$ and $L_j(k) = 0$ otherwise. Such polynomials always exist, and are called *Lagrange basis polynomials.*

For each $i \in [n]$, define $g_i(t,x_i) := \sum_{j\in[w]} L_j \cdot f_{j,i}(x_i)$, and let $g(t,\mathbf{x}) = \prod_{i\in[n]} g_i(t,x_i)$. Then $g(t = j,\mathbf{x}) = \prod_{i\in[n]} f_{j,i}(x_i)$, and hence $f(\mathbf{x}) = \sum_{j\in[w]} g(j,\mathbf{x})$ as required. ◀

4.1 An alternate view of commutative ROABPs

▶ **Definition 19.** *For an ideal $J \subset \mathbb{C}[\mathbf{t}]$, and a $G \in \mathbb{C}[\mathbf{t},\mathbf{x}]$ given by $G = \sum_{\mathbf{e}} \mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}}(G)(\mathbf{t})\cdot\mathbf{x}^{\mathbf{e}}$, we define the polynomial $\tilde{G} = (G \bmod J)$ as follows.*

$$\tilde{G} := \sum_{\mathbf{e}} (\mathrm{coeff}_{\mathbf{x}^{\mathbf{e}}}(G)(\mathbf{t}) \bmod J)\cdot\mathbf{x}^{\mathbf{e}}$$

Here $(g(\mathbf{t}) \bmod J)$ for any $g(\mathbf{t})$ is defined as per Definition 34.

Using the above definition, given any commutative ROABP, we can come up with a product of univariates over $\mathbf{x}$s that is related to it in the following sense.

▶ **Lemma 20.** *Suppose $f(\mathbf{x}) = \mathbf{b}^{\mathsf{T}}\left(\prod_{i\in[n]}\left(A_{i,0} + A_{i,1}x_i + \cdots + A_{i,d}x_i^d\right)\right)\mathbf{c}$, is a commutative-ROABP of width w computing $f(\mathbf{x})$.*

Then there exists an ideal $J \subset \mathbb{C}[t_1,\ldots,t_r]$ with a finite variety, and $G(\mathbf{t},\mathbf{x}) := \prod_i G_i(\mathbf{t},x_i)$, such that for $\tilde{G}(\mathbf{t},\mathbf{x}) := G(\mathbf{t},\mathbf{x}) \bmod J$, $f(\mathbf{x})$ can be expressed as a linear combination of the $\mathbf{t}$-coefficients of $\tilde{G}$.

Furthermore, $|\mathbf{t}| = r \leq \min\left\{w^2, n(d+1)\right\}$ and the $\mathbf{t}$-degree of each G_i is at most w^2.

Proof. Let $F(\mathbf{x})$ denote the $w \times w$ matrix with entries in $\mathbb{C}[\mathbf{x}]$, so that $f(\mathbf{x}) = \mathbf{b}^{\mathsf{T}}F(\mathbf{x})\mathbf{c}$. Let $\mathbf{A} = \{A_1,\ldots,A_r\}$ be such that the ring $\mathbb{C}[A_1,\ldots,A_r]$ is the same as that generated by the coefficient matrices $\{A_{i,j}\}$. It is easy to see that $r \leq \min\left\{w^2, n(d+1)\right\}$.

We define the ideal J as follows: $J = \{g(\mathbf{t}) \in \mathbb{C}[\mathbf{t}] : g(A_1, \ldots, A_r) = 0\}$. Let $N_J = \{\mathbf{t}^{\mathbf{a}_1}, \ldots, \mathbf{t}^{\mathbf{a}_m}\}$ be the *normal set* of J; then $|N_J| = m \leq w^2$, as the quotient ring of J is isomorphic to $\mathbb{C}[\mathbf{A}] \subset \mathbb{C}^{w\times w}$ (see Lemma 23). For each i, j let $G_{i,j}(\mathbf{t})$ be the polynomial with monomials from N_J such that $G_{i,j}(\mathbf{A}) = A_{i,j}$. We define $G_i(\mathbf{t}, x_i) = \sum_j G_{i,0}(\mathbf{t})x_i^j$ for each $i \in [n]$. Since N_J is closed under divisions, the degree of any $\mathbf{t}^{\mathbf{a}} \in N_J$ is at most w^2, and hence $\deg_{\mathbf{t}}(G_i) = \deg(G_{i,j}) \leq w^2$ for all i.

Let $\tilde{G} := (G \bmod J) = \sum_{\mathbf{a}\in N_J} \tilde{g}_{\mathbf{a}}(\mathbf{x})\mathbf{t}^{\mathbf{a}}$ for some $\tilde{g}_{\mathbf{a}}(\mathbf{x})$s, which we call the "**t**-coefficients of G".

$$\begin{aligned}
f(\mathbf{x}) &= \sum_{k,\ell\in[w]} b_k c_\ell \cdot F(\mathbf{x})[k,\ell] \\
\text{(By definition of } G) &= \sum_{k,\ell\in[w]} b_k c_\ell \cdot \left(G(\mathbf{A}, \mathbf{x})\right)[k,\ell] \\
\text{(By definition of } J) &= \sum_{k,\ell\in[w]} b_k c_\ell \cdot \left(\tilde{G}(\mathbf{A}, \mathbf{x})\right)[k,\ell] \\
\text{(Expanding } \tilde{G}) &= \sum_{k,\ell\in[w]} b_k c_\ell \cdot \left(\sum_{\mathbf{a}\in N_J} \tilde{g}_{\mathbf{a}}(\mathbf{x})\mathbf{A}^{\mathbf{a}}\right)[k,\ell] \\
\text{(For } A_{\mathbf{a}} = \mathbf{A}^{\mathbf{a}}) &= \sum_{\mathbf{a}\in N_J} \left(\sum_{k,\ell\in[w]} b_k c_\ell A_{\mathbf{a}}[k,\ell]\right) \tilde{g}_{\mathbf{a}}(\mathbf{x}) = \sum_{\mathbf{a}\in N_J} \beta_{\mathbf{a}}\tilde{g}_{\mathbf{a}}(\mathbf{x})
\end{aligned}$$

In the last line above, $A_{\mathbf{a}} \in \mathbb{F}^{w\times w}$ is the matrix that the "monomial" $\mathbf{A}^{\mathbf{a}}$ evaluates to. ◀

4.2 Evaluating derivatives of polynomials

We now show that for any polynomials $g(\mathbf{t})$, $h(\mathbf{t})$, and any point $\bar{\alpha} \in \mathbb{C}^r$, the value $(D_h(g))(\bar{\alpha})$ can be obtained as a linear combination of $O(d', \mathrm{WR}(h))$ evaluations of the polynomial g, where $d' = \max\{\deg(g), \deg(h)\}$. This is a known fact (see e.g. [16]). We only state the lemma here, and provide a proof in the full version.

We start with a fact about the "symmetry" between $D_h(g)(\bar{0})$ and $D_g(h)(\bar{0})$ that we will need.

▶ **Fact 4.1.** *For any $g, h \in \mathbb{C}[t_1, \ldots, t_r]$, $D_g(h)(\bar{0}) = D_h(g)(\bar{0}) = \sum_{\mathbf{e}\in\mathbb{N}^r} \mathbf{e}!g_{\mathbf{e}}h_{\mathbf{e}}$.*

▶ **Lemma 21** (Functionals and Waring rank). *Let $g, h \in \mathbb{C}[t_1, \ldots, t_r]$ be polynomials of degree at most d', and suppose $\mathrm{WR}(h) \leq s$. Then there exist $W = O(s \cdot d')$ points $\mathbf{y}_1, \ldots, \mathbf{y}_W$ such that $D_h(g)(\bar{0}) = D_g(h)(\bar{0})$ can be expressed as a linear combination of $g(\mathbf{y}_1), \ldots, g(\mathbf{y}_W)$.*

4.3 The proof

We now have all the pieces required to prove the main theorem, which we first restate.

▶ **Theorem 2.** *For any $n, r \in \mathbb{N}$, let $S(r, m)$ denote the smallest $\Sigma\wedge\Sigma$-size required to compute any r-variate polynomial f with $\mathrm{DPD}(f) \leq m$.*
Then for all $n, d, w \in \mathbb{N}$, $\mathsf{commROABP}(n, d, w) \subseteq \mathsf{diagROABP}\left(n, d, S(w^2, w^2)nw^4\right)$.

Proof. Let $F(\mathbf{x}) = \prod_{i=1}^n \left(\sum_{j=0}^d A_{i,j}x_i^j\right)$, and let $f(\mathbf{x}) = \mathbf{b}^{\intercal}F(\mathbf{x})\mathbf{c}$ be the corresponding commutative ROABP of width w.

Moving to the polynomial world: From Lemma 20, there is a $G(\mathbf{t},\mathbf{x}) = \prod_{i\in[n]} G_i(\mathbf{t},x_i)$ such that $f(\mathbf{x})$ is a linear combination of the $\mathbf{t}$-coefficients of $\tilde{G} := G \bmod J$, where J is the *ideal of dependencies* of the coefficient matrices $\{A_{i,j}\}$.
Let $r = |\mathbf{t}|$, $\mathbf{V}(J) = \{\bar{\alpha}_1,\ldots,\bar{\alpha}_z\}$, and $N_J = \mathrm{NS}(J)$ with $m = |N_J|$. Then $r, m \le w^2$ and $\deg_{\mathbf{t}}(G_i) \le w^2$ for all $i \in [n]$, and there exist $\beta_{\mathbf{a}}$s and $\tilde{g}_{\mathbf{a}}(\mathbf{x})$s such that

$$f(\mathbf{x}) = \sum_{\mathbf{a}\in N_J} \beta_{\mathbf{a}}\tilde{g}_{\mathbf{a}}(\mathbf{x}).$$

Coefficients from derivatives: Next, the results from [12, 13] (Lemma 36) imply that there exist m polynomials $\{h_{u,v}(\mathbf{t})\}$ such that:

- $\mathrm{DPD}(h_{u,v}) \le m$ for all $h_{u,v}$, and
- For any $\mathbf{a} \in N_J$, $\mathrm{coeff}_{\mathbf{a}}(\tilde{G}) = \sum_{u,v} \gamma^{\mathbf{a}}_{u,v}(D_{h_{u,v}}(G))(\bar{\alpha}_u)$, for some $\left\{\gamma^{\mathbf{a}}_{u,v}\right\} \subset \mathbb{C}$.

Derivatives using evaluations: Then, using Lemma 21 we see that for any polynomial h with $s := \mathrm{WR}(h)$ and for any polynomial G with $\deg(g), \deg(h) \le d'$, there exist at most $s \cdot d'$ points $\mathbf{y}_1,\ldots,\mathbf{y}_{sd'} \in \mathbb{C}^r$ and constants $\lambda_1,\ldots,\lambda_{sd'} \in \mathbb{C}$ such that :

$$(D_h(G))(\bar{\alpha}) = \sum_{q=1}^{sd'} \lambda_q G(\mathbf{y}_q).$$

Thus, for all u, v, $O(\mathrm{WR}(h_{u,v}) \cdot \max\{\deg_{\mathbf{t}}(G), \deg(h_{u,v})\}) = O(S(r,m)\cdot nw^2)$ evaluations of G are enough to obtain $(D_{h_{u,v}}(G))(\bar{\alpha}_u)$.

Putting everything together: Combining all the steps, we get the following.

$$\begin{aligned}
f(\mathbf{x}) &= \sum_{\mathbf{a}\in N_J} \beta_{\mathbf{a}}\tilde{g}_{\mathbf{a}}(\mathbf{x}) \\
&= \sum_{\mathbf{a}\in N_J} \beta_{\mathbf{a}} \sum_{u,v} \gamma^{\mathbf{a}}_{u,v}(D_{h_{u,v}}(G))(\bar{\alpha}_u) \\
\text{(Rearranging)} &= \sum_{u,v}\left(\sum_{\mathbf{a}\in N_J} \beta_{\mathbf{a}}\gamma^{\mathbf{a}}_{u,v}\right)(D_{h_{u,v}}(G))(\bar{\alpha}_u) \\
\text{(For appropriate } \beta'\text{s)} &= \sum_{u,v} \beta'_{u,v}(D_{h_{u,v}}(G))(\bar{\alpha}_u) \\
\left(\mathrm{DPD}(h_{u,v}) \le m,\ \deg(G) \le nw^2\right) &= \sum_{u,v} \beta'_{u,v} \sum_{q=1}^{S(r,m)\cdot nw^2} \lambda_q G(\mathbf{y}_q,\mathbf{x}) \\
\therefore f(\mathbf{x}) &= \sum_{q'=1}^{m\cdot S(r,m)\cdot nw^2} \mu_{q'} \prod_{i\in[n]} G_i(\mathbf{y}_q, x_i)
\end{aligned}$$

Thus, as $m, r \le w^2$, we get a diagonal ROABP for $f(\mathbf{x})$ of width $O(w^2 \cdot S(w^2,w^2)\cdot nw^2) = O(S(w^2,w^2)\cdot nw^4)$. ◀

▶ Remark 22. Suppose there exists an explicit polynomial f that is computable by a commutative ROABP of polynomial width but any diagonal ROABP computing f requires width super-polynomial in n. Let w be the width of the commutative ROABP, and let J be the ideal of dependencies of its coefficient matrices. By Lemma 36 there exist polynomials $\{h_{u,v}(\mathbf{t})\}$ with $|\mathbf{t}| \le w^2$, such that $\mathrm{DPD}(h_{u,v}) \le w^2$. But if $\mathrm{WR}(h_{u,v}) = \mathrm{poly}(w)$ for each u, v, then we should get a diagonal ROABP of width $\mathrm{poly}(w)$, which is a contradiction. Thus, a separation between commutative and diagonal ROABPs also leads to an explicit polynomial that witnesses the separation dimension of partial derivatives and Waring rank.

References

1 Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983. doi:10.1016/0304-3975(83)90110-X.

2 Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. A quadratic lower bound for algebraic branching programs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.2.

3 David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties and Algorithms*. Undergraduate texts in mathematics. Springer, 2007. doi:10.1007/978-0-387-35651-8.

4 David S. Dummit and Richard M. Foote. *Abstract Algebra*. John Wiley and Sons, Inc., second edition, 1999.

5 Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Demystifying the border of depth-3 algebraic circuits. In *62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, 2021. URL: https://www.cse.iitk.ac.in/users/nitin/papers/border-depth3.pdf.

6 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. doi:10.1109/FOCS.2013.34.

7 Ignacio García-Marco, Pascal Koiran, Timothée Pecatte, and Stéphan Thomassé. On the complexity of partial derivatives. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 37:1–37:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.37.

8 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and commutative, read-once oblivious abps. *Theory of Computing*, 13(1):1–21, 2017. doi:10.4086/toc.2017.v013a002.

9 Neeraj Kayal. Affine projections of polynomials. In *44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 643–662, 2012. doi:10.1145/2213977.2214036.

10 Mrinal Kumar and Ben Lee Volk. A lower bound on determinantal complexity. In *36th Annual Computational Complexity Conference (CCC 2021)*, volume 200 of *LIPIcs*, pages 4:1–4:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.4.

11 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. *Electron. Colloquium Comput. Complex.*, page 81, 2021. URL: https://eccc.weizmann.ac.il/report/2021/081.

12 M.G. Marinari, H.M. Möller, and T. Mora. Gröbner bases of ideals defined by functionals with an application to ideals of projective points. *Applicable Algebra in Engineering, Communication and Computing*, 4(2):103–145, 1993. doi:10.1007/BF01386834.

13 H. Michael Möller and Hans J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik*, 70, 1995. doi:10.1007/s002110050122.

14 Noam Nisan. Lower bounds for non-commutative computation. In *23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418, 1991. doi:10.1145/103418.103462.

15 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. doi:10.1007/BF01294256.

16 Kevin Pratt. Waring rank, parameterized and exact algorithms. In David Zuckerman, editor, *60th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2019)*, pages 806–823. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00053.

17 Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6(1):135–177, 2010. doi:10.4086/toc.2010.v006a007.

18 Chandan Saha. Factoring Polynomials over Finite Fields using Balance Test. In *25th Symposium on Theoretical Aspects of Computer Science (STACS 2008)*, pages 609–620, 2008.

19 Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015. URL: `https://github.com/dasarpmar/lowerbounds-survey/releases/`.

20 Yaroslav Shitov. How hard is the tensor rank?, 2016. `arXiv:1611.01559`.

21 Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. `doi:10.1007/PL00001609`.

22 V. Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, 13(3):354–356, 1969. `doi:10.1007/BF02165411`.

23 Leslie G. Valiant. Completeness Classes in Algebra. In *11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 249–261, 1979. `doi:10.1145/800135.804419`.

A Algebraic structure of commutative ROABPs

This section is aimed at equipping the reader with the algebraic-geometric concepts about *rings generated by commuting matrices*, that are required to understand the results in [12] and [13] (Lemma 32 and Lemma 36). It is therefore largely expository, and readers who are comfortable with these concepts may skip it.

We start by analysing rings generated by a single matrix in Subsection A.1, and then extend our observations to general rings of matrices in Subsection A.2.

A.1 Rings generated by a single matrix

For any matrix $A \in \mathbb{C}^{w\times w}$, the commutative ring generated by A that is denoted by $\mathbb{C}[A]$, is the set of all matrices that can be written as univariate polynomials in terms of A. In other words, $\mathbb{C}[A] := \{p(A) : p(t) \in \mathbb{C}[t]\}$.

Observe that the matrices $I(= A^0), A, A^2, \ldots, A^w$ satisfy the linear dependency that is given by the *characteristic polynomial of* A: $\det(A - tI) \in \mathbb{C}[t]$. Thus, $\mathbb{C}[A]$ is a vector space (over $\mathbb{C}$) of dimension at most w.

In fact the dimension of $\mathbb{C}[A]$ could be even smaller, and it is captured by the degree of the *minimal polynomial of* A: the smallest degree polynomial $p(t)$ such that $p(A)$ is the zero matrix; and the ideal generated by p, $\langle p\rangle := \{q(t) \in \mathbb{C}[t] : q(t) \text{ is divisible by } p(t)\}$, characterises the ring $\mathbb{C}[A]$. The following fact formalises this relationship.

▶ **Fact A.1.** *Let $A \in \mathbb{C}^{w\times w}$ and let $p(t) \in \mathbb{C}[t]$ be its minimal polynomial. Then the ring generated by A, $\mathbb{C}[A]$, is* isomorphic *to the quotient ring* $\mathbb{C}[t]/\langle p\rangle$.

Proof. Define $\Phi : \mathbb{C}[t] \to \mathbb{C}[A]$ such that $\Phi(q(t)) = q(A)$ for any q. Then the following facts together show that the restriction of Φ on $\mathbb{C}[t]/\langle p\rangle$ is a ring isomorphism by the *first ring isomorphism theorem* (see e.g. [4]).

- Φ is a ring homomorphism: $\Phi(q_1 + q_2 \cdot q_3) = (q_1 + q_2 \cdot q_3)(A) = q_1(A) + q_2(A) \cdot q_3(A)$.
- Φ is *onto*: Trivially follows from the definition of $\mathbb{C}[A]$.
- $\ker \Phi = \langle p\rangle$: Suppose $\Phi(q) = 0$. Then $q(A) = 0$, which implies that $q(t) = p(t) \cdot q'(t)$ as $p(t)$ is the minimal polynomial of A. ◀

Let us now focus on the quotient ring of the ideal generated by an arbitrary polynomial $p(t)$; we shall later rephrase our findings in terms of matrices.

Suppose $p(t) = (t - \alpha_1)^{e_1}(t - \alpha_2)^{e_2} \cdots (t - \alpha_z)^{e_z}$, of degree $m = \sum_u e_u$. Since we are working over $\mathbb{C}$, this is true without loss of generality. Let p_u be the polynomial $(t - \alpha_u)^{e_u}$, for each $u \in [z]$. Then any polynomial $q(t)$ is divisible by p_u whenever α_u is a root of $q(t)$ and its first $(e_u - 1)$ derivatives. In fact, $q(t)$ is divisible by $p = \prod_u p_u$, exactly when the above condition holds for each $u \in [z]$.

▶ **Fact A.2.** *A polynomial $q(t)$ is divisible by $p(t) = \prod_{u\in[z]}(t-\alpha_u)^{e_u}$ if and only if:*

$$\forall u \in [z], \qquad q(\alpha_u) = \frac{\partial q}{\partial t}(\alpha_u) = \frac{\partial^2 q}{\partial t^2}(\alpha_u) = \cdots = \frac{\partial^{e_u-1} q}{\partial t^{e_u-1}}(\alpha_u) = 0.$$

In other words, the $\sum_u e_u = m$ values obtained by evaluating the appropriate derivatives of q at the corresponding roots of p, tell us whether p divides q. These evaluations of derivatives in fact give us some more information about q with respect to the ideal $\langle p\rangle$, which we now see.

Derivatives characterise the quotient ring. For any polynomials $p(t), q(t)$ we define the "remainder polynomial" $q(t) \bmod p(t)$ as follows.

$$q(t) \bmod p(t) = \tilde{q}(t), \text{ such that } q(t) = q'(t)p(t) + \tilde{q}(t), \text{ with } \deg(\tilde{q}) < \deg(p)$$

Suppose $p(t)$ is a polynomial of degree m, then $\tilde{q}(t)$ is clearly a polynomial of degree at most $m-1$. It turns out that the d evaluations of derivatives of q given in Fact A.2 completely determine $\tilde{q}$.

▶ **Fact A.3.** *Suppose $p(t) = \prod_{u\in[z]}(t-\alpha_u)^{e_u}$ has degree m, then there exist m^2 constants $\left\{\gamma^a_{u,v}\right\} \subset \mathbb{C}$ such that for any polynomial $q(t)$, we have*

$$\forall 0 \le a \le m-1, \qquad \tilde{q}_a = \sum_{\substack{u\in[z]\\ v\in[e_u]}} \gamma^{(a)}_{u,v} \cdot \frac{\partial^v q}{\partial t^v}(\alpha_u),$$

where $\tilde{q}(t) := \sum_{0\le j\le m-1} \tilde{q}_j t^j = q(t) \bmod p(t)$.

A.2 General commutative matrix rings

The above observations about "univariate" rings can be summarised as follows. Firstly, any matrix ring is isomorphic to the quotient ring of an ideal, where this ideal contains all polynomial dependencies that the generator matrix satisfies (Fact A.1); thus every matrix in the ring corresponds to a polynomial modulo this ideal.
Secondly, the remainder of any polynomial q with respect to this ideal is completely determined by the evaluations of certain derivatives of q at appropriate points (Fact A.3).

We shall now see the multivariate analogues of the above facts, which tell us about rings generated by multiple commuting matrices.

To fix some notation, suppose that we have been given the $w \times w$ matrices $A_1, \ldots, A_r$ that all commute with each other. These matrices therefore generate a commutative ring of matrices denoted by $\mathbb{C}[A_1, \ldots, A_r]$, whose algebraic properties we shall now provide.

A.2.1 Matrix rings as quotient rings of ideals

Recall that for the ring $\mathbb{C}[A]$, the corresponding ideal was $\langle p(t)\rangle$, where p was the minimal polynomial of A. The ideal $\langle p(t)\rangle$ precisely contains all the polynomials $q(t)$ for which $q(A) = 0$. Therefore a natural choice for the multivariate ideal is the *ideal of dependencies of* $A_1, \ldots, A_r$, $J := \{q(t_1, \ldots, t_r) \in \mathbb{C}[\mathbf{t}] : q(A_1, \ldots, A_r) = 0\}$. Indeed, the quotient ring of J is isomorphic to $\mathbb{C}[A_1, \ldots, A_r]$.

▶ **Lemma 23.** *Suppose $A_1, A_2, \ldots, A_r \in \mathbb{C}^{w\times w}$ are mutually commutative, and let J be their* ideal of dependencies *inside the r-variate polynomial ring $\mathbb{C}[\mathbf{t}]$. Then $\mathbb{C}[A_1, \ldots, A_r]$ is isomorphic to ${}^{\mathbb{C}[\mathbf{t}]}/_{J}$.*

Proof. Similar to the proof of Fact A.1, we define the map $\Phi : \mathbb{C}[\mathbf{t}] \to \mathbb{C}[A_1, \ldots, A_r]$, which maps $q(\mathbf{t})$ to the matrix $q(A_1, \ldots, A_r)$. This naturally defines the (restricted) map $\phi : {}^{\mathbb{C}[\mathbf{t}]}/_{J} \to \mathbb{C}[A_1, \ldots, A_r]$, with $\phi(\tilde{q}) = \tilde{q}(\mathbf{A})$.

The following facts are now easy to verify for Φ, which together prove that ϕ is an isomorphism by the *first ring isomorphism theorem* (see e.g. [4]).

- Φ is a ring homomorphism: $\Phi(q_1+q_2\cdot q_3) = (q_1+q_2\cdot q_3)(A_1, \ldots, A_r) = q_1(\mathbf{A})+q_2(\mathbf{A})\cdot q_3(\mathbf{A}) = \Phi(q_1) + \Phi(q_2) \cdot \Phi(q_3)$.
- Φ is *onto*: Trivially follows from the definition of $\mathbb{C}[\mathbf{A}]$.
- $\ker \Phi = J$: Suppose $\Phi(q) = 0$. Then $q(\mathbf{A}) = 0$, which implies that $q(\mathbf{t}) \in J$. ◀

We note an important property of the ideal J, before moving on to the next part. Notice that the minimal polynomials of each of the matrices $A_1, \ldots, A_r$, say $p_1(t_1), p_2(t_2), \ldots, p_r(t_r)$ are elements of J. This means that J contains univariate polynomials in each of its underlying variables. Thus, the set of common zeroes of polynomials in J, also known as the *variety of* J(denoted by $\mathbf{V}(J)$), is finite. One way to see this is that $\mathbf{V}(J) \subseteq \text{roots}(p_1) \times \text{roots}(p_2) \times \cdots \times \text{roots}(p_r)$, where $\text{roots}(p_i)$ denotes the constants in $\mathbb{C}$ where p_i vanishes, and × denotes the *Cartesian product* of sets. Such ideals are called *zero dimensional ideals*, because their variety is a zero dimensional set in the ambient space $\mathbb{C}^r$.

▶ **Definition 24** (Zero-dimensional ideals). *An ideal $J \subseteq \mathbb{C}[\mathbf{t}]$ is called* zero-dimensional *if its variety is finite; i.e. $|\mathbf{V}(J)| < \infty$.*

A.2.2 Quotient rings of zero dimensional ideals

Since we are interested in zero dimensional ideals J, we shall now assume that $\mathbf{V}(J) = \{v_1, \ldots, v_z\}$ for some $z \in \mathbb{N}$.

Arguably, the statements we have discussed till this point are fairly well-known. But we believe that most of the ideas we shall now see are not as commonly known, especially in the theoretical computer science community. We remark that much of the non-trivial ideas and proofs in this section (Appendix A) belong to previous works [12, 13].

Taking a cue from Fact A.3, for a zero-dimensional ideal J we expect the "multiplicities" of the points in its variety $\mathbf{V}(J)$ to help us find the correct derivatives. In this case, the commonly used definition of multiplicity for multivariate polynomials: multiplicity of w means *all* partial derivatives of order $< w$ vanish, turns out to be a little too coarse. In order to formally introduce the suitable definition, we need the following notion of *derivative operators*, which are like polynomials whose monomials are partial derivatives.

▶ **Definition 25** (Derivative operators). *A* derivative operator *on $\mathbb{C}[t_1, \ldots, t_r]$ is a $\mathbb{C}$-linear combination of* finitely many *partial derivatives of the form $\partial_{\mathbf{a}} : \mathbb{C}[\mathbf{t}] \to \mathbb{C}[\mathbf{t}]$, where $\mathbf{a} \in \mathbb{N}^r$.*

The operator $D = \sum_{\mathbf{a}} \gamma_{\mathbf{a}} \partial_{\mathbf{a}}$ naturally maps a polynomial $q(\mathbf{t}) \in \mathbb{C}[\mathbf{t}]$, to $(\sum_{\mathbf{a}} \gamma_{\mathbf{a}} \cdot \partial_{\mathbf{a}} q(\mathbf{t}))$ which we denote by $D(q)$.

Any polynomial $h(\mathbf{t})$ naturally defines a derivative operator $D_h := \sum_{\mathbf{a} \in \text{supp}(h)} \text{coeff}_h(\mathbf{a}) \partial_{\mathbf{a}}$. Likewise, one can talk about the polynomial that underlies a derivative operator.

In Fact A.3, the set of derivative-evaluations that characterise the ideal generated by a $p = (t - \alpha)^e$, are evaluations at α of derivatives with respect to the monomials $\{t^{e-1}, t^{e-2}, \ldots, t, 1\}$; for multiple factors we take the union of the evaluations for each factor. In particular, there is a "maximum" derivative ${}^{\partial^e}/_{\partial t^e}$, and the other derivatives are obtained by "down-shifting" it (similar to taking all possible derivatives of the underlying monomial). This observation leads us to define the following notion of *shifts* of derivatives and derivative operators.

▶ **Definition 26** (Shifts of derivatives and derivative operators). *For a partial derivative $\partial_{\mathbf{e}} : \mathbb{C}[\mathbf{t}] \to \mathbb{C}[\mathbf{t}]$ and a vector $\mathbf{a} \geq \bar{0}$, we define the* **a**-shift of $\partial_{\mathbf{e}}$, *denoted by $\sigma_{\mathbf{a}}(\partial_{\mathbf{e}})$, as follows.*

$$\sigma_{\mathbf{a}}(\partial_{\mathbf{e}}) := \begin{cases} \frac{\mathbf{e}!}{(\mathbf{e}-\mathbf{a})!} \cdot \partial_{\mathbf{e}-\mathbf{a}} & \text{if } \mathbf{a} \leq \mathbf{e}, \\ 0 & \text{otherwise.} \end{cases}$$

The definition naturally extends to **a**-shift of D_h, *denoted by $\sigma_{\mathbf{a}}(D_h)$, as follows.*

$$\sigma_{\mathbf{a}}(D_h) := \sum_{\mathbf{e}:\mathbf{e}\geq\mathbf{a}} \text{coeff}_{\mathbf{e}}(h) \cdot \sigma_{\mathbf{a}}(\partial_{\mathbf{e}}) = \sum_{\mathbf{e}:\mathbf{e}\geq\mathbf{a}} \text{coeff}_{\mathbf{e}}(h) \cdot \frac{\mathbf{e}!}{(\mathbf{e}-\mathbf{a})!} \cdot \partial_{\mathbf{e}-\mathbf{a}}$$

The following observations about derivative operators and their shifts will be useful.

▶ **Observation 27.** *For any derivative operator D_h and vector* **a**, $\sigma_{\mathbf{a}}(D_h) = D_{\partial_{\mathbf{a}}(h)}$.

▶ **Observation 28.** *For any derivative operator D_h and polynomials $p(\mathbf{t}), q(\mathbf{t})$, we have the following.*

$$D_h(p \cdot q) = \sum_{\mathbf{a}} \frac{1}{\mathbf{a}!} \cdot \partial_{\mathbf{a}}(p) \cdot \sigma_{\mathbf{a}}(D_h)(q) = \sum_{\mathbf{a}} \frac{1}{\mathbf{a}!} \cdot \partial_{\mathbf{a}}(p) \cdot D_{\partial_{\mathbf{a}}(h)}(q)$$

In the language of shifts of derivative operators, we can say that the set of derivatives with respect to $\{t^e, t^{e-1}, \ldots, t, 1\}$ is *down-closed*: closed under taking shifts. The following definitions then follow naturally.

▶ **Definition 29** (Down-closed spaces of derivative operators). *A $\mathbb{C}$-vector space of derivative operators Δ is said to be* down-closed *if for all $D \in \Delta$, any shift D' of D, also belongs to Δ.*

▶ **Definition 30** (Closure of an operator). *For a polynomial $h(t_1, \ldots, t_r) \in \mathbb{C}[\mathbf{t}]$ and the corresponding derivative operator D_h, we define the* closure of D_h *as follows.*
$\Delta(h) := \{D_{\partial_{\mathbf{e}}(h)} : \mathbf{e} \in \mathbb{N}^r, \partial_{\mathbf{e}}(h) \neq 0\}$.

Ideals with a single point in their variety and closed spaces of derivative operators have the following interesting connection, similar to a univariate ideal $\langle (t-\alpha)^e \rangle$.

▶ **Lemma 31.** *Let $J \in \mathbb{C}[t_1, \ldots, t_r]$ be an ideal with $\mathbf{V}(J) = \{\bar{\alpha}\}$, then the set $\Delta(J)$ of derivative operators defined by $\Delta(J) := \{D \in \mathbb{C}[\partial t_1, \ldots, \partial t_r] : \forall g \in J, D(g)(\bar{\alpha}) = 0\}$ a closed vector space.*

Proof. Firstly, for all D_1, D_2, and $\beta \in \mathbb{C}$, $(\beta D_1 + D_2)(f)(\bar{\alpha}) = \beta D_1(f)(\bar{\alpha}) + D_2(f)(\bar{\alpha}) = 0$, just by linearity of differentiation. So $\Delta(J)$ is a vector space over $\mathbb{C}$.

To see that it is closed, suppose $D_h \in \Delta(J)$ for a polynomial $h(\mathbf{t})$, and let $i \in [r]$ be such that the partial derivative $h' := {}^{\partial h}/_{\partial t_i} \neq 0$. Then using Observation 28, for any $g \in J$ we have that $D_h(t_i \cdot g)(\bar{\alpha}) = (t_i \cdot D_h(g) + 1 \cdot D_{h'}(g))(\bar{\alpha}) = v_i \cdot D_h(g)(\bar{\alpha}) + 1 \cdot D_{h'}(g)(\bar{\alpha})$. Now since J is an ideal, $g \in J$ implies that $t_i \cdot g \in I$ and therefore $D_h(t_i \cdot g)(\bar{\alpha}) = 0$; and $D_h(g)(\bar{\alpha}) = 0$ because $g \in J$ and $D_h \in \Delta(J)$. Thus, $D_{h'}(g)(\bar{\alpha}) = 0$ for any $D_h \in \Delta(J)$ and $i \in [r]$ such that ${}^{\partial h}/_{\partial t_i} \neq 0$. The closure under an arbitrary shift **a** then follows by induction on the **a**. ◀

We are now ready to state the following result which follows from the work of Marinari, Möller and Mora [12, Theorem 2.6], which is a suitable multivariate analogue for Fact A.2.

▶ **Lemma 32** (Zero dimensional ideals and derivative operator spaces). *Suppose an ideal $J \subseteq \mathbb{C}[\mathbf{t}]$ has variety $\mathbf{V}(J) = \{\bar{\alpha}_1, \ldots, \bar{\alpha}_z\}$ and $\dim_{\mathbb{C}} ({}^{\mathbb{C}[\mathbf{t}]}/_J) = m$. Then there exist closed spaces of derivative operators $\Delta_1, \ldots, \Delta_z$ of dimensions $m_1, \ldots, m_z$ with $\sum_u m_u = m$, such that for any polynomial $g(\mathbf{t}) \in \mathbb{C}[\mathbf{t}]$ we have that $g \in J$, if and only if $\forall u \in [z], \forall D \in \Delta_u : D(g)(\bar{\alpha}_u) = 0$.*

Thus, every zero-dimensional ideal is characterised by a set of closed spaces of derivative operators, where the number of spaces is equal to the size of the variety. Next, we see how one can obtain "$g \bmod J$" given the $\sum_u m_u = m$ derivative-evaluations corresponding to the z bases of $\Delta_1, \ldots, \Delta_z$. To that end, we first formalise what $g \bmod J$ means and then state a result from [13] that provides the above solution.

A.2.3 Matrices and polynomials in the quotient ring

When dealing with univariate polynomials, it is quite straightforward to define $q(t) \bmod p(t)$ as $r(t)$, such that $q(t) = q'(t)p(t) + r(t)$ for some polynomial $q'(t)$ with $\deg(r) < \deg(p)$. This is because we intuitively identify $r(t)$ to be "less than" $p(t)$ since it has smaller degree, and thus the concepts of division and remainders extend naturally. However, things are a little more tricky for multivariate polynomials: e.g. which monomial is "smaller"? x^2 or y^2?

We therefore need to fix a consistent way of comparing any two given monomials; we need a *monomial ordering*: a total ordering on monomials that "respects" division/multiplication (see e.g. [3, Chapter 2]). We shall skip the formal definition of a monomial ordering, and just work with the "dictionary ordering" or *lexicographic ordering*: $\mathbf{t}^{\mathbf{a}} \prec \mathbf{t}^{\mathbf{a}'}$ if the smallest $i \in [r]$ with $a_i \neq a'_i$ is such that $a_i < a'_i$. Using the monomial ordering $\prec$, we can define the *leading monomial* of a polynomial, and then *leading monomials of J* for an ideal J.

▶ **Definition 33** (Leading monomials). *For a polynomial $g(\mathbf{t})$, a monomial $\mathbf{t}^{\mathbf{a}} \in \operatorname{supp}(g)$ is said to be the* leading monomial *of g, denoted by $\operatorname{LM}(g)$, if for all $\mathbf{t}^{\mathbf{a}'} \in \operatorname{supp}(g)$ we have that $\mathbf{t}^{\mathbf{a}'} \prec \mathbf{t}^{\mathbf{a}}$.*

Similarly, we define $\operatorname{LM}(J) := \{\operatorname{LM}(g) : g \in J\}$ for an ideal J.

We can then define the remainder of a polynomial with respect to an ideal J.

▶ **Definition 34** (Remainder modulo an ideal). *For a polynomial $g(\mathbf{t})$ and an ideal $J \subset \mathbb{C}[\mathbf{t}]$, we say that $g(\mathbf{t}) \bmod J = \tilde{g}(\mathbf{t})$, if there exist polynomials $g_J(\mathbf{t}) \in J$ and $\tilde{g}(\mathbf{t})$ such that $g(\mathbf{t}) = g_J(\mathbf{t}) + \tilde{g}(\mathbf{t})$, where $\operatorname{LM}(\tilde{g})$ does not belong to the ideal $\langle \operatorname{LM}(J) \rangle$.*

Observe that if $LM(\tilde{g}) \notin \langle \operatorname{LM}(J) \rangle$, then in fact no monomial in $\operatorname{supp}(\tilde{g})$ belongs to the ideal $\langle \operatorname{LM}(J) \rangle$. And thus $\operatorname{supp}(\tilde{g})$ is contained in the "complement of $\langle \operatorname{LM}(J) \rangle$", called the *normal set of J*.

▶ **Definition 35** (Normal set of an ideal). *For an ideal $J \in \mathbb{C}[t_1, \ldots, t_r]$, the* normal set *of J is defined as $\operatorname{NS}(J) := \{\mathbf{t}^{\mathbf{a}} : \mathbf{a} \in \mathbb{N}^r, \mathbf{t}^{\mathbf{a}} \notin \langle \operatorname{LM}(J) \rangle\}$.*

We sometimes overload notation to denote $\operatorname{NS}(J)$ as the set of exponent vectors. That is, $\operatorname{NS}(J) = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ means $\operatorname{NS}(J) = \{\mathbf{t}^{\mathbf{a}_1}, \ldots, \mathbf{t}^{\mathbf{a}_m}\}$.

Here are some important properties of the normal set of an ideal (see e.g. [12]).

▶ **Fact A.4.** *For any ideal J, its normal set $\operatorname{NS}(J)$ has the following properties.*

- *For any $g(\mathbf{t})$, the polynomial $g \bmod J$ is a linear combination of monomials in $\operatorname{NS}(J)$, and further, $|\operatorname{NS}(J)| = \dim_{\mathbb{C}} \left(\mathbb{C}[\mathbf{t}]/J\right)$.*
- *$\operatorname{NS}(J)$ is closed under divisions. That is, if $\mathbf{t}^{\mathbf{a}} \in \operatorname{NS}(J)$ and $\mathbf{t}^{\mathbf{a}'} | \mathbf{t}^{\mathbf{a}}$, then $\mathbf{t}^{\mathbf{a}'} \in \operatorname{NS}(J)$. In particular, $1 \in N_J$ for all ideals J.*

We can now state the result of Möller and Stetter [13] that gives a more explicit version of the correspondence in Lemma 32. The following is a multivariate analogue of Fact A.3.

▶ **Lemma 36** (Consequence of [13, Theorem 1]). *Suppose $J \subset \mathbb{C}[t_1, \dots, t_r]$ is an ideal with variety $\mathbf{V}(J) = \{\bar{\alpha}_1, \dots, \bar{\alpha}_z\}$ and normal set $N_J := \mathrm{NS}(J) = \{\mathbf{a}_1, \dots, \mathbf{a}_w\}$. Let $\Delta_1, \dots, \Delta_z$ be the characterising derivative operator spaces, with each Δ_u spanned by $\{D_{u,1}, \dots, D_{u,m_u}\}$, such that $|N_J| = m = \sum_u m_u$.*

Then there exists a set of m^2 constants $\left\{\gamma_{u,v}^{(\mathbf{a})}\right\} \subset \mathbb{C}$, such that for any polynomial $g(\mathbf{t}) \in \mathbb{C}[\mathbf{t}]$ and $\tilde{g}(\mathbf{t}) := (g(\mathbf{t}) \bmod J)$, we have $\mathrm{coeff}_{\mathbf{a}}(\tilde{g}) = \sum_{u,v} \gamma_{u,v}^{(\mathbf{a})}(D_{u,v}(g))(\bar{\alpha}_u)$ for all $\mathbf{a} \in N_J$.

A Relativization Perspective on Meta-Complexity

Hanlin Ren
University of Oxford, UK

Rahul Santhanam
University of Oxford, UK

Abstract

Meta-complexity studies the complexity of computational problems about complexity theory, such as the Minimum Circuit Size Problem (MCSP) and its variants. We show that a relativization barrier applies to many important open questions in meta-complexity. We give relativized worlds where:

1. MCSP can be solved in deterministic polynomial time, but the search version of MCSP cannot be solved in deterministic polynomial time, even approximately. In contrast, Carmosino, Impagliazzo, Kabanets, Kolokolova [CCC'16] gave a randomized approximate search-to-decision reduction for MCSP with a relativizing proof.
2. The complexities of MCSP$[2^{n/2}]$ and MCSP$[2^{n/4}]$ are different, in both worst-case and average-case settings. Thus the complexity of MCSP is not "robust" to the choice of the size function.
3. Levin's time-bounded Kolmogorov complexity $\mathrm{Kt}(x)$ can be approximated to a factor $(2+\epsilon)$ in polynomial time, for any $\epsilon > 0$.
4. Natural proofs do not exist, and neither do auxiliary-input one-way functions. In contrast, Santhanam [ITCS'20] gave a relativizing proof that the non-existence of natural proofs implies the existence of one-way functions under a conjecture about optimal hitting sets.
5. DistNP does not reduce to GapMINKT by a family of "robust" reductions. This presents a technical barrier for solving a question of Hirahara [FOCS'20].

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Circuit complexity; Theory of computation → Problems, reductions and completeness

Keywords and phrases meta-complexity, relativization, minimum circuit size problem

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.54

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2021/089` [40]

1 Introduction

Meta-complexity refers to the complexity of computing complexity. A prominent example of a meta-complexity problem is the Minimum Circuit Size Problem (MCSP): Given as input the (length-2^n) truth table of a function $f : \{0,1\}^n \to \{0,1\}$, output the size of the smallest circuit that computes f. MCSP was recognized as a fundamental problem in the Soviet Union since 1950s [43], and has received a lot of attention in the last two decades since the seminal work of Kabanets and Cai [28]. Other examples include computing variants of Kolmogorov complexity such as polynomial-time bounded Kolmogorov complexity and Levin's time-bounded Kolmogorov complexity Kt [2, 29]. Questions about the circuit size of Boolean functions are closely related to Kolmogorov complexity and incompressibility, because a circuit is essentially a *compressed representation* of the truth table of the function it computes.

There has been plenty of interplay between meta-complexity and other areas of complexity theory such as average-case complexity [15, 16, 18, 19], cryptography [32, 38, 39, 42], learning theory [10, 36] and pseudorandomness [2, 17, 28, 36].

We highlight a couple of recent breakthrough results. The first gives a non-black-box worst-case to average-case reduction for a problem about Kolmogorov complexity ("GapMINKT") that many believe to be NP-hard.

▶ **Theorem 1** ([15], building on [10]). *There is a randomized polynomial-time worst-case to average-case reduction for* GapMINKT.

The second gives an *equivalence* between the existence of one-way functions and the bounded-error average-case hardness over the uniform distribution of the functional version of MINKT. This result *characterizes* the most fundamental primitive in cryptography by a notion in meta-complexity.

▶ **Theorem 2** ([32]). *One-way functions exist if and only if there is a polynomial p such that the $p(n)$-time bounded Kolmogorov complexity of a string x of length n cannot be computed in polynomial time on average, when x is chosen uniformly at random from n-bit strings.*

Results such as these give hope for a rich theory connecting complexity lower bounds, meta-complexity, average-case complexity, learning theory and cryptography, among other fields. However, despite much effort, many basic questions about meta-complexity remain elusive. In addition, the recent advances on meta-complexity also propose new questions, some of which are seemingly beyond our reach. (See Section 1.1 for a sample of these questions.)

In this work, we seek a more fine-grained understanding of the current landscape of meta-complexity by using the classical perspective of *relativization* [9]. It is noteworthy that Theorem 1 and Theorem 2 relativize. Of course, we need to be careful here to define what relativization means, as the notion typically applies to complexity classes and not to computational problems. However, meta-computational problems do indeed have natural notions of relativizations, where the algorithms solving the problem as well as the algorithms defining the problem get access to the same oracle A. Results such as Theorem 1 and Theorem 2 use techniques from the theory of pseudorandomness [27, 34, 44], which typically relativize, and it is worth asking how much these techniques can achieve. Can they be used to solve the major open problems in the area?

We give a largely negative answer to this question, by giving oracles relative to which many of the questions in the area have answers opposite to what we expect. However, we do not necessarily infer that there are fundamental barriers to solving the major open questions; we can only say that new techniques will be required in many cases. Our perspective also contributes to formulating new notions and questions which might still be approachable using current techniques. We also note that there are some exciting recent works in meta-complexity by Ilango and others (e.g. [22–25]) using gate elimination and related ideas. It is not clear yet whether relativization is a barrier to these techniques.

1.1 Our Questions

We first introduce the questions with which we are concerned.

1.1.1 Easiness or Hardness of Meta-Complexity Problems

Arguably, the most important and fundamental problem about MCSP is whether MCSP is easy or hard. Is MCSP in polynomial time, or if not, is MCSP NP-complete? It is reported in [5, 30] that Levin delayed the publication of his NP-completeness results [31] because he wanted to show NP-hardness for MCSP. A long line of research [3, 4, 12, 20, 21, 28, 33, 41] showed that the NP-completeness of MCSP implies breakthrough results in complexity theory. For instance, if MCSP is NP-complete under polynomial-time Karp reductions, then EXP $\neq$ ZPP [33]. However, these results do not indicate whether MCSP is or is not NP-complete; they merely suggest that this problem will be hard to solve.

▶ **Question 3.** *Is* MCSP NP*-complete under polynomial-time Karp reductions?*

Just as with MCSP, it is open to show the NP-hardness of MINKT. A further motivation for this problem is the recent "non-black-box" worst-case to average-case reduction for MINKT [15]. As a consequence, if GapMINKT is NP-hard, then the worst-case and average-case complexities of NP are equivalent. As there are serious obstacles to showing the NP-completeness of MINKT by "weak" reductions, [15] proposed, as a weakening of Question 3, that MINKT could be NP-hard via very powerful reductions:

▶ **Question 4.** *Is* GapMINKT NP*-hard under* $\mathsf{coNP}_{/\mathrm{poly}}$*-Turing reductions?*

In terms of unconditional lower bounds, there is an intriguing question about the meta-complexity of Levin's Kt complexity, raised in [2]. It is known that MKtP is EXP-complete, but only under rather powerful reductions such as $\mathsf{P}_{/\mathrm{poly}}$-truth-table reductions or NP-Turing reductions. Therefore, it is reasonable to conjecture that MKtP is not in P. However, the aforementioned reducibilities are too strong, so we cannot apply the time hierarchy theorem directly to prove that MKtP $\notin$ P. Still, it may be surprising that this problem has been open for almost 20 years:[1]

▶ **Question 5.** *Is* MKtP *computable (or at least approximable) in polynomial-time?*

(We note that a randomized version of MKtP, called MrKtP, is known to be not in BPP unconditionally [35].)

1.1.2 Structural Properties of Meta-Complexity Problems

Every NP-complete problem admits a *search-to-decision* reduction. For instance, given an oracle that decides SAT, for every input formula φ that is satisfiable, we can find a satisfying assignment of φ in polynomial time. However, it is unknown whether MCSP has this property.

▶ **Question 6.** *Does* MCSP *admit a search-to-decision reduction?*

We remark that there has been some progress on Question 6: [10] showed that if MCSP is in BPP, then a certain "weak" version of search-MCSP can be solved in probabilistic polynomial time; [23] presented a "non-trivial" search-to-decision reduction for the problem of minimizing formulas.

Another mystery about MCSP is whether its various *parameterized* versions are equivalent. Specifically, let MCSP$[s(n)]$ denote the problem that given a truth table of a function $f : \{0,1\}^n \to \{0,1\}$, determine whether f can be computed by a circuit of size $s(n)$. It is easy to see that MCSP$[2^{n/2}]$ reduces to MCSP$[2^{n/4}]$,[2] but the converse direction is unknown:

▶ **Question 7.** *Is* MCSP$[2^{n/4}]$ *reducible to* MCSP$[2^{n/2}]$ *under polynomial-time Karp reductions?*

The average-case version of Question 7 is also open. It is observed in [19] that any errorless heuristic for MCSP$[2^{n/2}]$ can be transformed into an errorless heuristic for MCSP$[2^{n/4}]$, but the converse is unknown.

[1] The conference version of [2] was published in 2002.

[2] Given an input truth table f of length 2^n, let f' be the concatenation of 2^n copies of f, then $f' : \{0,1\}^{2n} \to \{0,1\}$ is a function that only depends on half of its input bits, and the circuit complexities of f and f' are exactly the same. Therefore $f \in$ MCSP$[2^{n/2}]$ if and only if $f' \in$ MCSP$[2^{n/4}]$.

▶ **Question 8.** *If* MCSP$[2^{n/4}]$ *is easy on average, does this imply that* MCSP$[2^{n/2}]$ *is also easy on average?*

One drawback of the worst-case to average-case reduction of [15] is that it only works for *zero-error* average-case complexity. Ideally, we would like to establish a worst-case to *two-sided-error* average-case reduction for MINKT. Can we extend the results in [15] to the two-sided-error setting?

▶ **Question 9.** *Is there a natural distribution such that, if* MINKT *is easy on this distribution with two-sided error, then* GapMINKT *is solvable in the worst case? In particular, does the* uniform *distribution satisfy the above condition?*

1.1.3 Meta-Complexity, Average-Case Complexity and Cryptography

Some of the most compelling questions around meta-complexity relate to connections with average-case complexity and cryptography. A partial converse of [15] was established in [16,17], where it was shown that if GapMINKT$^{\text{SAT}} \in \mathsf{P}$, then $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, i.e. NP is easy on average. Here GapMINKT$^{\text{SAT}}$ is the problem of determining the (time-bounded) Kolmogorov complexity of a string with a SAT oracle. Based on this result, [16] characterized the average-case complexity of the polynomial hierarchy by the worst-case complexity of meta-complexity. An important open question, a positive answer to which would imply a characterization of the average-case complexity for NP, is whether the SAT oracle can be removed, that is:

▶ **Question 10.** *Does* GapMINKT $\in \mathsf{P}$ *imply* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*?*

There seems to be strong correspondences between the hardness of MCSP and problems in cryptography. For example, if MCSP is easy, then one-way functions (OWFs) do not exist [28, 38]. Under the unproven Universality Conjecture, [42] established the converse direction, i.e. if MCSP is zero-error average-case hard, then OWFs exist. Of course, an *unconditional* answer would be much more interesting:

▶ **Question 11.** *Can we base the existence of OWF from the nonexistence of natural proofs?*

A recent exciting work [32] established the equivalence between the two-sided error average-case hardness of MINKT and the existence of one-way functions. Given the result in [32], it is perhaps natural to conjecture that GapMINKT $\in \mathsf{CZK}$ unconditionally, where CZK is the set of languages with a computational zero-knowledge proof system [14]. One could imagine a win-win argument as follows: If MINKT is easy, then of course it is in CZK; on the other hand, if MINKT is hard, then one-way functions exist, and by the result of [14], every language in NP is in CZK. However, there are some gaps between the "easy" and "hard" in the above argument, as we do not know what happens if MINKT is only worst-case hard and one-way functions do not exist.

▶ **Question 12.** *Does (some gap version of)* MCSP *or* MINKT *admit a computational zero knowledge proof system?*

2 Our Results

In this work, we investigate the above questions in the perspective of *relativization*. Due to page limits, we only describe our results in this section and provide a proof overview in Section 3. The detailed proofs can be found in the full version of this paper [40].

2.1 Meta-Complexity Problems Are Not Robust in Relativized Worlds

In our first set of results, we present evidence for the following hypothesis: A *slight change* in the definition of a meta-complexity problem could result in a *completely different* problem. For example, we show that there are relativized worlds where MCSP is significantly easier than search-MCSP, and relativized worlds where MCSP$[2^{n/2}]$ and MCSP$[2^{n/4}]$ have dramatically different complexities.

▶ **Theorem 13** (Informal version). *For each of the following items, there is a relativized world where it becomes true.*

- MCSP $\in \mathsf{P}$*, but* search-MCSP *is very hard.*
- MCSP$[2^{n/2}] \in \mathsf{P}$*, but* MCSP$[2^{n/4}]$ *is very hard.*
- MCSP$[2^{n/4}]$ *admits a polynomial-time errorless heuristic, but* MCSP$[2^{n/2}]$ *does not.*

As direct consequences of Theorem 13, we have the following nonreducibility results: For example, unless nonrelativizing techniques are used, MCSP does not admit a search-to-decision reduction, and MCSP$[2^{n/4}]$ does not reduce to MCSP$[2^{n/2}]$.

2.2 Barriers for Proving Hardness of Kt Complexity

Our second result concerns Question 5.

▶ **Theorem 14** (Informal version). *There is a relativized world where Levin's* Kt *complexity can be* $(2+\epsilon)$*-approximated in polynomial time.*

We note that Question 5 also appeared in a stronger form in literature. In particular, let R_{Kt} be the set of strings x such that $\mathrm{Kt}(x) \geq |x|/3$, it is conjectured that any "dense enough" subset of R_{Kt} is not in polynomial time. Our result shows that this conjecture needs nonrelativizing techniques to prove.

Actually, our message is even stronger than the above statement of Theorem 14. We define a nonstandard variant of Levin's Kt complexity, and denote it as $\widetilde{\mathrm{Kt}}$, such that $\widetilde{\mathrm{Kt}}$ approximates Kt, i.e. for every string x, $\widetilde{\mathrm{Kt}}(x) \leq \mathrm{Kt}(x) \leq (2+o(1))\widetilde{\mathrm{Kt}}(x)$. Then we construct a relativized world where $\widetilde{\mathrm{Kt}}$ is computable in polynomial time *exactly*, and Theorem 14 follows directly.

However, non-relativizing techniques already play an important role in characterizing the complexity of R_{Kt}. It was shown that any dense subset of R_{Kt} is EXP-complete under $\mathsf{P}_{/\mathrm{poly}}$-truth-table reductions and NP-Turing reductions [2], and these results use the non-relativizing "instance checkers" for EXP-complete problems [7, 8]. An *algebrization* barrier would be more satisfying for showing limitations of such techniques. However, we could not extend our oracle world to an algebrizing one in the sense of either [1], [26], or [6].

Nevertheless, we managed to construct an oracle world where $\widetilde{\mathrm{Kt}}$ is computable in polynomial time, and $\mathsf{EXP} = \mathsf{ZPP}$ holds simultaneously.

▶ **Theorem 15.** *There is a relativized world where* $\widetilde{\mathrm{Kt}}$ *complexity is computable in deterministic polynomial time, and* $\mathsf{EXP} = \mathsf{ZPP}$.

In this world, EXP-complete problems have trivial instance checkers, since they are in ZPP. We also get some other non-relativizing theorems such as $\mathsf{IP} = \mathsf{PSPACE}$ for free, since $\mathsf{PSPACE} \subseteq \mathsf{EXP} = \mathsf{ZPP} \subseteq \mathsf{IP}$. As a result, we cannot prove that $\widetilde{\mathrm{Kt}}$ is not in polynomial time, even if we combine $\mathsf{IP} = \mathsf{PSPACE}$ or the instance checkers for EXP-complete problems with relativizing techniques. We believe that this oracle world serves as a "fundamental obstacle" ([2]) to proving MKtP $\notin \mathsf{P}$.

We think our new complexity measure $\widetilde{\mathrm{Kt}}$ is of independent interest. Understanding $\widetilde{\mathrm{Kt}}$ using nonrelativizing techniques may serve as the first step towards solving Question 5.

2.3 Natural Proofs Versus Cryptography

Our third set of results is motivated by Question 11. Under the so-called "Universality Conjecture", [42] answered Question 11 affirmatively, i.e. the non-existence of natural proofs is equivalent to the existence of one-way functions. In contrast, we show that the answer of Question 11 is false in some relativized world, establishing a barrier for constructing one-way functions from nonexistence of natural proofs. We can even rule out *auxiliary-input* one-way functions (a primitive weaker than one-way functions) in our world.

Consequently, the Universality Conjecture fails in this world. As we will discuss in Section 3.3, in this world, the Universality Conjecture actually fails in a *very intuitive way.*

▶ **Theorem 16** (informal version)**.** *There is a relativized world where* $\mathsf{P}_{/\mathrm{poly}}$*-natural properties useful against* $\mathsf{SIZE}[2^{\delta n}]$ *do not exist, and auxiliary-input one-way functions do not exist either.*

The non-existence of natural proofs corresponds to the zero-error average-case hardness of MCSP [19]. We also extend our results by showing a relativized world where MCSP or MINKT is hard even for two-sided error heuristics.

▶ **Theorem 17** (informal version)**.** *There is a relativized world where* GapMCSP *is hard on average under some samplable distribution, and auxiliary-input one-way functions do not exist.*

▶ **Theorem 18** (informal version)**.** *There is a relativized world where* GapMINKT *is hard on average under some samplable distribution, and auxiliary-input one-way functions do not exist.*

Besides Question 11, we also show the following consequences based on our relativized worlds:

- (Question 9) Extending the results in [15] to the bounded-error case requires nonrelativizing techniques, if the underlying distribution for MINKT is still the uniform distribution. (This is because [32] showed the equivalence between the existence of one-way functions and the bounded-error average-case hardness of MINKT under the uniform distribution.)
- (Question 12) It requires nonrelativizing techniques to show that GapMINKT $\in \mathsf{CZK}$, or even that GapMINKT can be solved on average by a CZK protocol, on infinitely many input lengths. This is because [37] showed that if auxiliary-input one-way functions do not exist, then $\mathsf{CZK} = \mathsf{BPP}$.

  Note that the proof that if one-way functions exist then $\mathsf{NP} \subseteq \mathsf{CZK}$ [14] is already nonrelativizing. On the other hand, we show that basing GapMINKT $\in \mathsf{CZK}$ on the *non*existence of one-way functions also requires a nonrelativizing proof.

2.4 Limits of GapMINKT as an Oracle

We also present technical barriers for showing *stronger* reductions to the GapMINKT oracle, such as coNP-Turing reductions or $\mathsf{P}_{/\mathrm{poly}}$-Turing reductions.

We view (Turing) reductions to a promise problem $L = (L.\textsc{Yes}, L.\textsc{No})$ as machines that interact with an (adversarial) oracle, and tries to solve a problem L'. We say a reduction is *robust*, if it works even if the adversary is *inconsistent* on queries not in the promise. That is, on queries outside ($L.\textsc{Yes} \cup L.\textsc{No}$), the adversary can sometimes return 0 and sometimes return 1. Furthermore, the adversary is allowed to see the input of L' or the nondeterministic branch the reduction is running on, and decide whether to return 0 or 1 accordingly.

We show that a reduction that is both robust and relativizing cannot solve Question 10 or (a harder version of) Question 4. However, as the requirement of robust reductions seem very strong, we mainly treat these results as *technical* barriers rather than *conceptual* barriers. It is also worth mentioning that we use the "Gap" in GapMINKT in a very crucial way.

▶ **Theorem 19** (informal version). *Each of the following items* cannot *be proved by a reduction that is both robust and relativizing.*

- *Either* GapMINKT $\in$ coNP*, or* GapMINKT *is* NP*-complete under* coNP*-Turing reductions.*
- *Every problem in* DistNP *has a polynomial-size two-sided error heuristic with* GapMINKT *oracles.*

We did not manage to prove non-hardness results under $\mathsf{coNP}_{/\mathrm{poly}}$-Turing reductions, as mentioned in Question 4. We leave it as an open problem.

▶ **Open Problem 20.** *Is there a relativized world where* GapMINKT $\notin \mathsf{coNP}_{/\mathrm{poly}}$*, and* GapMINKT *is not* NP*-complete under robust* $\mathsf{coNP}_{/\mathrm{poly}}$*-Turing reductions?*

3 Technical Overview

3.1 Meta-Complexity Problems Are Not Robust in Relativized Worlds

We briefly discuss the proof techniques of the first bullet of Theorem 13 here, i.e. there is an oracle world such that MCSP is easy but search-MCSP is hard. The framework for the other two bullets will be similar.

Making MCSP easy. We can add an MCSP oracle in our oracle world, but the circuit minimization problem in our world becomes $\mathrm{MCSP}^{\mathrm{MCSP}}$. Then we also need to add an $\mathrm{MCSP}^{\mathrm{MCSP}}$ oracle, but again, the circuit minimization problem becomes $\mathrm{MCSP}^{\mathrm{MCSP}^{\mathrm{MCSP}}}$ now. Therefore, a natural approach is to add the "limit" of

$$\mathrm{MCSP}^{\mathrm{MCSP}^{\mathrm{MCSP}^{\cdots}}}$$

into our oracle world. Indeed, this is what we do: We add an oracle itrMCSP (which stands for "iterated MCSP") into our world, such that (roughly speaking)

$$\mathrm{itrMCSP}[k, x, s] = \underbrace{\mathrm{MCSP}^{\mathrm{MCSP}^{\mathrm{MCSP}^{\cdots}}}}_{\text{iterate } k \text{ times}}[x, s].$$

(Recall that $\mathrm{MCSP}^{\mathcal{O}}[x, s] = 1$ if and only if in the oracle world with oracle $\mathcal{O}$, the circuit complexity of the truth table x is at most s.)

In our world, MCSP is indeed easy. Actually, let x be a truth table of length 2^n, then the circuit complexity of x is at most s in our world if and only if $\mathrm{itrMCSP}[2^n, x, s] = 1$.

Making search-MCSP hard. We define an oracle $\mathcal{O}$ that diagonalizes against every polynomial time Turing machine M, and define itrMCSP relative to $\mathcal{O}$. (That is, for example, $\mathrm{itrMCSP}[1, x, s] = \mathrm{MCSP}^{\mathcal{O}}[x, s]$ and $\mathrm{itrMCSP}[2, x, s] = \mathrm{MCSP}^{\mathrm{MCSP}^{\mathcal{O}}}[x, s]$.) For every Turing machine M, we find a large enough integer N and a hard truth table x_{hard}

of length $\mathrm{poly}(N)$. Then we feed x_{hard} to M. How we answer the $\mathcal{O}$ queries of M is not important, but each time M makes a query $\mathrm{itrMCSP}[k, x, s]$, we *pretend x has the lowest possible circuit complexity*, and answer this query accordingly.

To be more precise, we fix the oracle $\mathcal{O}$ up to input length $N - 1$ before we simulate M on input x_{hard}. This has the effect that for every integer k, truth table x, and parameter $s \le N - 1$, we already know whether $\mathrm{itrMCSP}[k, x, s] = 1$ regardless of how we fix the rest of $\mathcal{O}$; see Claim 3.3 of the full version. Then upon every query $\mathrm{itrMCSP}[k, x, s]$, if $s \le N - 1$ we already know how to reply to it; otherwise we simply reply 1.

At last, for every query $\mathrm{itrMCSP}[k, x, s]$ where $s \ge N$ and we returned 1, we need to put the truth table x in the length-N slice of $\mathcal{O}$ so that its circuit complexity is indeed at most N. Since M only runs in polynomial time, and only probes very few positions of $\mathcal{O}$, we can indeed put it somewhere in $\mathcal{O}$ without letting M notice. We do not need to care about the parameter k here, as $\mathrm{MCSP}[x, N] = 1$ implies $\mathrm{itrMCSP}[k, x, N] = 1$ for every k.[3] To diagonalize against M, we also put x_{hard} into the length-N slice of $\mathcal{O}$, but in a place that M did not probe at all. In this way, we can guarantee that there is a size-N circuit for x_{hard}, but M fails to find it.

3.2 Barriers for Proving Hardness of Kt Complexity

We first define the complexity $\widetilde{\mathrm{Kt}}$. For a string x, let $\widetilde{\mathrm{Kt}}(x)$ denote the minimum possible value of $|M| + \lfloor \log t \rfloor$, where after we run the machine M on the empty input for t steps, the content of some tape of M is exactly x. The difference between Kt and $\widetilde{\mathrm{Kt}}$ is that in the definition of Kt, we require M to halt after outputting x; while in the definition of $\widetilde{\mathrm{Kt}}$, x can be an intermediate step of the computation.

A fixed-point oracle. Our approach will be to find a "fixed-point" of $\widetilde{\mathrm{Kt}}$: an oracle $\mathcal{O}$ such that $\mathcal{O}[x] = \widetilde{\mathrm{Kt}}^{\mathcal{O}}(x)$ for every string x. Then, in the world with oracle $\mathcal{O}$, we can compute $\widetilde{\mathrm{Kt}}(x)$ by simply calling $\mathcal{O}[x]$.

We proceed in stages, and in stage n, we fix the strings that have $\widetilde{\mathrm{Kt}}$ complexity exactly n. We enumerate every (M, t) such that $|M| + \lfloor \log t \rfloor = n$, and run M for t steps. For every intermediate tape content x, if $\mathcal{O}[x]$ is not fixed yet, then we fix $\mathcal{O}[x] = n$. A natural problem is: how to respond to the $\mathcal{O}$ queries made by M? The answer is surprisingly simple: for every query $\mathcal{O}[y]$ that M makes, we already have $\widetilde{\mathrm{Kt}}(y) \le n$ by definition, so if $\mathcal{O}[y]$ is not fixed to a value smaller than n yet, then we can return $\mathcal{O}[y] = n$ confidently! It is not hard to show that the oracle $\mathcal{O}$ is indeed a "fixed-point" of $\widetilde{\mathrm{Kt}}$.

Achieving $\mathsf{EXP} = \mathsf{ZPP}$. It is also simple to achieve $\mathsf{EXP} = \mathsf{ZPP}$ in the above oracle. To simulate exponential time, we give the zero-error probabilistic polynomial-time machine a "cheat" oracle Cheat that embeds the truth tables of a certain EXP-complete problem. It is natural to choose the EXP-complete problem as

$$L = \{(M, t) : M \text{ on empty input outputs 1 in time } t\},$$

since we can construct $\mathcal{O}$ and obtain the truth tables of L at the same time. We can reply arbitrarily when M queries the Cheat oracle.

Now we have a "fixed-point" oracle $\mathcal{O}$ such that $\mathcal{O}[x] = \widetilde{\mathrm{Kt}}^{\mathcal{O},\mathsf{Cheat}}(x)$ for every x. We also have a length-2^n truth table (of L), which we want to "embed" into Cheat. We can simply embed it into the length-$3n$ (say) slice of Cheat, as there are still many empty slots

[3] It is possible to define itrMCSP such that this is satisfied.

not asked in the construction of $\mathcal{O}$. Actually, the number of empty slots is so large (around $2^{3n} - 2^n\text{poly}(n)$) that we can embed it "everywhere we can". A ZPP algorithm can simply guess a pointer in the length-$3n$ slice of Cheat, and it will likely point to the truth table of L.

3.3 Natural Proofs Versus Cryptography

We only discuss how we prove Theorem 16. Our starting point is an oracle world in [45, Section 5], in which there is a hard-on-average problem but no auxiliary-input one-way functions. Given a function $f : \{0,1\}^n \to \{0,1\}^n$ (think of f as a uniformly random function), the world consists of two oracles: A PSPACE-complete oracle, and a "verification" oracle for f:

$$V_f[x, y] = \begin{cases} 1 & \text{if } f(x) = y, \\ 0 & \text{otherwise.} \end{cases}$$

Inverting auxiliary-input one-way functions. We use essentially the same argument as in [45]. Roughly speaking, given any circuit C of size s, it is possible to "eliminate" every V_f gate in C, and obtain a circuit C' of size poly(s), such that C and C' agree on a $1 - 1/s$ fraction of inputs, but C' does not use V_f at all. This is because V_f behaves like an oracle that is both random and sparse. Therefore, for each V_f gate, we only need to store its answers to the inputs that appear frequently, and V_f is likely zero on other inputs.

Now, given any circuit C, we want to "invert" C, i.e. given $C(\mathbf{z})$ for a uniformly random input $\mathbf{z}$, output any string in $C^{-1}(C(\mathbf{z}))$. We simply find a circuit C' that is close to C, uses no V_f gates, and is only polynomially larger than C. Then we use the PSPACE-complete oracle to invert C'.

Ruling out natural proofs. It suffices to show there is a *succinct pseudorandom distribution*, i.e. a distribution $\mathcal{D}$ over truth tables with small circuits, such that $\mathcal{D}$ is indistinguishable from the uniform distribution by small circuits. (Actually, this approach is inspired by recent circuit lower bounds [11, 19] for MCSP.)

Let $\mathcal{D}$ be any distribution over poly(s) strings, that fools PSPACE-oracle circuits of size s. The existence of $\mathcal{D}$ can be proven by the probabilistic method. For each $x \in \{0,1\}^{O(\log s)}$, let D_x be the x-th truth table in $\mathcal{D}$. We "embed" D_x into the oracle $V_f[x, f(x)]$, as follows:

$$V_f[x, y, \beta] = \begin{cases} D_x[\beta] & \text{if } f(x) = y, \\ \perp & \text{otherwise.} \end{cases}$$

Here, $D_x[\beta]$ is the β-th bit of D_x. Now we have artificially made $\mathcal{D}$ a *succinct* distribution: the circuit complexity of every string in $\mathcal{D}$ is small. We also need to prove $\mathcal{D}$ is *pseudorandom*, i.e. it fools every size $s^{o(1)}$ circuit. For every circuit C with V_f gates and PSPACE gates, we use the same method as above to eliminate every V_f gate in C, to obtain a circuit C' that is close to C. Note that the distribution under which we measure the closeness of C and C' is a hybrid of $\mathcal{D}$ and the uniform distribution. After that, we can use the fact that $\mathcal{D}$ fools C' to also show that $\mathcal{D}$ fools C, therefore C cannot be a natural proof.

How did the Universality Conjecture fail? The Universality Conjecture of [42] roughly says that if there are succinct pseudorandom distributions, then there are *efficiently samplable* succinct pseudorandom distributions. However, in our oracle world, the succinct pseudorandom distribution $\mathcal{D}$ does not appear to be efficiently samplable: to sample from $\mathcal{D}$, it seems that we need be able to compute f, which is hard when f is a random function.

3.4 Limits of GapMINKT as an Oracle

At the core of our proofs is the following weakness of GapMINKT: *It may hide a small change of the oracle.* In particular, suppose we have two oracles $\mathcal{O}$ and $\mathcal{O}'$, such that they only differ at one input, then the "Gap" in GapMINKT allows us to choose an instantiation of GapMINKT that is both consistent with $\text{GapMINKT}^{\mathcal{O}}$ and $\text{GapMINKT}^{\mathcal{O}'}$. (See Lemma 6.2 in the full version.) This instantiation of GapMINKT would not help the reduction distinguish between $\mathcal{O}$ and $\mathcal{O}'$ at all; however, an NP problem on $\mathcal{O}$ and $\mathcal{O}'$ may have very different answers.

NP-intermediateness under coNP-Turing reductions. It is not hard to construct a relativized world where GapMINKT $\notin$ coNP (see, e.g. [29, Theorem 4.1]). For the "non-completeness" part, we construct a diagonalizing oracle $\mathcal{O}$ such that there is no robust reduction from the NP problem

$$L = \{0^n : \mathcal{O} \cap \{0,1\}^n \neq \varnothing\}$$

to GapMINKT. On input length N, we construct a GapMINKT oracle that is both consistent with "$\mathcal{O} \cap \{0,1\}^N = \varnothing$" and "$|\mathcal{O} \cap \{0,1\}^N| = 1$". This oracle does not reveal whether $0^N \in L$, and we can still use the standard method to diagonalize against every co-nondeterministic Turing machine. In particular, we run this machine and reply 0 to all its queries to $\mathcal{O}$. If it rejects some branch, we put a string of length N that is not probed in this branch into $\mathcal{O}$; otherwise we do nothing.

Non-DistNP-hardness under $\mathsf{P}_{/\mathsf{poly}}$-Turing reductions. [13] showed that a random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ cannot be computed on average by circuits of size $2^{o(n)}$, even with a verification oracle

$$\Pi[\alpha, \beta] = \begin{cases} 1 & \text{if } \pi(\alpha) = \beta, \\ 0 & \text{otherwise.} \end{cases}$$

We show the same thing for (robust) circuits with Π and GapMINKT oracle gates. To oversimplify, the argument boils down to the following task: Given an input α, a circuit C that computes π correctly on α, and every value $\{\pi(\beta)\}_{\beta \neq \alpha}$, recover $\pi(\alpha)$. Without GapMINKT gates, it suffices to use $\log |C|$ bits to store a number k, such that on input α, the k-th Π gate of C contains the correct answer $\pi(\alpha)$. (For comparison, the trivial solution needs to record $n \gg \log |C|$ bits.)

Now, the circuit C has GapMINKT gates, and it is robust in the sense that $C^{\Pi,B}(\alpha) = \pi(\alpha)$ for *every* oracle B consistent with GapMINKT. Now we let B' be the MINKT oracle in the world where $\Pi[\alpha, \pi(\alpha)] = 0$, and other entries of Π are not changed. As the new oracle Π does not depend on $\pi(\alpha)$ at all, we can simulate $C^{\Pi,B'}(\alpha)$ without knowing $\pi(\alpha)$. On the other hand, we only modified one entry in Π, therefore B' is still consistent with GapMINKT. We still record the number k defined above for the simulation $C^{\Pi,B'}(\alpha)$, which suffices to recover $\pi(\alpha)$.

4 Related Works

In the paper that defined MINKT, Ko [29] studied the properties of MINKT in relativized worlds. Among other results, [29] showed that there is a relativized world where MINKT is neither in coNP, nor NP-complete under polynomial-time Turing reductions. This result indicates that the MINKT counterpart of Question 3 cannot be shown affirmatively using relativizing techniques. Also, [29] constructed a relativized world where $\mathsf{NP} \neq \mathsf{coNP}$, but MINKT *is* NP-complete under coNP-Turing reductions ("$\leq_T^{\mathsf{SNP}}$-reductions"). This leads to the conjecture [15, 29] that MINKT might be NP-complete under coNP-Turing reductions in the unrelativized world (Question 4).

Our third set of results build upon the results of Wee [45]. The motivation of [45] was to show that a certain cryptographic object (succinct noninteractive argument, SNARG) does not imply one-way functions in a relativizing way. The framework of [45] was very helpful for us, as we also need to rule out (auxiliary-input) one-way functions.

Xiao [46] presented a relativized world where learning is hard against circuits and auxiliary-input one-way functions do not exist either. It may seem that our results are direct corollaries of this result, since [10] proved that natural proofs imply learning algorithms. However, [46] only ruled out learning algorithms that use *uniform samples*, while the learning algorithms in [10] need *membership queries*. It seems that our results and [46] are incomparable. However, we remark that the techniques underlying [45, 46] and our results are quite similar.

We also mention the negative results of Hirahara and Watanabe [20] that has a different but similar setting compared to ours. In particular, they consider reductions to MCSP (in the unrelativized world) that are *oracle-independent*, i.e. work for MCSP^A for every oracle A. Two particular results in [20] are that deterministic oracle-independent reductions cannot reduce problems outside P to MCSP, and that randomized oracle-independent reductions that only make one query cannot reduce problems outside $\mathsf{AM} \cap \mathsf{coAM}$ to MCSP. As discussed in [20], the difference between relativization and their model is that in the relativized world with A oracle, a Turing reduction has access to not only MCSP^A but also A itself; however in their model, the reduction does not have access to A.

References

1 Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, 2009. doi:10.1145/1490270.1490272.

2 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal of Computing*, 35(6):1467–1493, 2006. doi:10.1137/050628994.

3 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616.

4 Eric Allender, Rahul Ilango, and Neekon Vafa. The non-hardness of approximating circuit size. In *Proc. 14th International Computer Science Symposium in Russia (CSR)* , volume 11532 of *Lecture Notes in Computer Science*, pages 13–24, 2019. doi:10.1007/978-3-030-19955-5_2.

5 Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14–40, 2011. doi:10.1016/j.jcss.2010.06.004.

6 Barış Aydınlıoğlu and Eric Bach. Affine relativization: Unifying the algebrization and relativization barriers. *ACM Transactions on Computation Theory*, 10(1):1:1–1:67, 2018. doi:10.1145/3170704.

7 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. doi:10.1007/BF01200056.

8 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless $\mathsf{EXPTIME}$ has publishable proofs. *Computatioanl Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.

9 Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the $\mathsf{P} =?\mathsf{NP}$ question. *SIAM Journal of Computing*, 4(4):431–442, 1975. doi:10.1137/0204037.

10 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10.

11 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *ACM Transactions on Computation Theory*, 12(3):21:1–21:27, 2020. doi:10.1145/3404860.

12 Bin Fu. Hardness of sparse sets and minimal circuit size problem. In *Proc. 26th International Computing and Combinatorics Conference (COCOON)* , volume 12273 of *Lecture Notes in Computer Science*, pages 484–495, 2020. doi:10.1007/978-3-030-58150-3_39.

13 Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal of Computing*, 35(1):217–246, 2005. doi:10.1137/S0097539704443276.

14 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. doi:10.1145/116825.116852.

15 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018. doi:10.1109/FOCS.2018.00032.

16 Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020. doi:10.1109/FOCS46700.2020.00014.

17 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:10.1145/3357713.3384251.

18 Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 292–302, 2021. doi:10.1145/3406325.3451065.

19 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPIcs*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

20 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

21 John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Proc. 35th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 45 of *LIPIcs*, pages 236–245, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236.

22 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and $\mathsf{AC}^0[p]$. In *Proc. 11th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 151 of *LIPIcs*, pages 34:1–34:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.34.

23 Rahul Ilango. Connecting perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 31:1–31:35, 2020. doi:10.4230/LIPIcs.CCC.2020.31.

24 Rahul Ilango. Constant depth formula and partial function versions of MCSP are hard. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020. doi:10.1109/FOCS46700.2020.00047.

25 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 22:1–22:36, 2020. doi:10.4230/LIPIcs.CCC.2020.22.

26 Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 695–704, 2009. doi:10.1145/1536414.1536509.

27 Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.

28 Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.

29 Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM Journal of Computing*, 20(5):962–986, 1991. doi:10.1137/0220059.

30 Leonid A. Levin. Hardness of search problems. Accessed 12-June-2021. URL: https://www.cs.bu.edu/fac/lnd/research/hard.htm.

31 Leonid A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

32 Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118.

33 Cody D. Murray and R. Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004.

34 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

35 Igor Carboni Oliveira. Randomness and intractability in Kolmogorov complexity. In *Proc. 46th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 132 of *LIPIcs*, pages 32:1–32:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.32.

36 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPIcs*, pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18.

37 Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proc. Second Israel Symposium on Theory of Computing Systems, (ISTCS)*, pages 3–17, 1993. doi:10.1109/ISTCS.1993.253489.

38 Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

39 Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *Proc. 36th Computational Complexity Conference (CCC)*, volume 200 of *LIPIcs*, pages 35:1–35:58, 2021. doi:10.4230/LIPIcs.CCC.2021.35.

40 Hanlin Ren and Rahul Santhanam. A relativization perspective on meta-complexity. *Electron. Colloquium Comput. Complex.*, page 89, 2021. URL: https://eccc.weizmann.ac.il/report/2021/089.

41 Michael Saks and Rahul Santhanam. Circuit lower bounds from NP-hardness of MCSP under Turing reductions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 26:1–26:13, 2020. doi:10.4230/LIPIcs.CCC.2020.26.

42 Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In *Proc. 11th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 151 of *LIPIcs*, pages 68:1–68:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.68.

43 Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

44 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

45 Hoeteck Wee. Finding Pessiland. In *Proc. 3rd Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 429–442, 2006. doi:10.1007/11681878_22.

46 David Xiao. On basing ZK $\neq$ BPP on the hardness of PAC learning. In *Proc. 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 304–315, 2009. doi:10.1109/CCC.2009.11.

Superlinear Lower Bounds Based on ETH

András Z. Salamon

School of Computer Science, University of St Andrews, UK

Michael Wehar

Computer Science Department, Swarthmore College, PA, USA

Abstract

We introduce techniques for proving superlinear conditional lower bounds for polynomial time problems. In particular, we show that CircuitSAT for circuits with m gates and $\log(m)$ inputs (denoted by log-CircuitSAT) is not decidable in essentially-linear time unless the exponential time hypothesis (ETH) is false and k-Clique is decidable in essentially-linear time in terms of the graph's size for all fixed k. Such conditional lower bounds have previously only been demonstrated relative to the strong exponential time hypothesis (SETH). Our results therefore offer significant progress towards proving unconditional superlinear time complexity lower bounds for natural problems in polynomial time.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Complexity classes

Keywords and phrases Circuit Satisfiability, Conditional Lower Bounds, Exponential Time Hypothesis, Limited Nondeterminism

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.55

Related Version *Previous Version*: `https://arxiv.org/abs/2008.06805`

Funding *András Z. Salamon*: EPSRC grants EP/P015638/1 and EP/V027182/1.

Acknowledgements We greatly appreciate the help and suggestions that we received. We are especially grateful to Kenneth Regan and Jonathan Buss who shared a manuscript [9] on speed-up results relating time and space. We also thank Michael Fischer, Mike Paterson, and Nick Pippenger, who tracked down two manuscripts related to circuit simulations. In addition, we thank Karl Bringmann whose advice helped us to better align this work with recent advances in fine-grained complexity. Likewise, we recognize helpful discussions with Henning Fernau and all of the participants at the workshop on Modern Aspects of Complexity Within Formal Languages (sponsored by DFG). Finally, we very much appreciate all of the feedback from Joseph Swernofsky, suggestions from Ryan Williams, and comments from anonymous referees.

1 Introduction

1.1 Motivation

Developing a deeper understanding of polynomial time problems is essential to the fields of algorithm design and computational complexity theory. In this work, we build on prior concepts from the topic of limited nondeterminism to show a new kind of conditional lower bound for polynomial time problems where a small runtime improvement for one problem would lead to a substantial runtime improvement for another.

We proceed by introducing basic notions and explaining how they relate to existing work. A polynomial time problem is a decision problem that can be decided in $O(n^k)$ time for some constant k, where n denotes the input length. As usual, P denotes the class of polynomial time problems. A decision problem has an unconditional time complexity lower bound $t(n)$ if it cannot be decided in $o(t(n))$ time. Polynomial time problems with unconditional superlinear time complexity lower bounds do not commonly appear in complexity theory research (aside from problems with lower bounds based on restrictive models such as one-tape Turing machines [24, 34]). Such problems are known to exist by the deterministic time hierarchy theorem [23], but to the best of our knowledge, there are few examples that appear in the literature. Most of the known examples are related to pebbling games [3] or intersection non-emptiness for automata [39, 41]. For these examples, the unconditional lower bounds are proven by combining Turing machine simulations with classical diagonalization arguments.

Although unconditional lower bounds are rare, many polynomial time problems have been shown to have conditional lower bounds in recent works on fine-grained complexity theory (see surveys [45, 7]). Our primary goal is to introduce superlinear conditional lower bounds based on weaker hypotheses than existing works, by applying new relationships between deterministic and nondeterministic computations.

1.2 Our Contribution

In this work, all logarithms are base 2 and we say that a problem is solvable in essentially-linear time if it is decidable in $O(n^{1+\varepsilon})$ time for all $\varepsilon > 0$.

The log-CircuitSAT decision problem (previously investigated in [8, 2]) is a natural restriction of circuit satisfiability to bounded fan-in Boolean circuits with m gates and $\log(m)$ inputs. Like many problems in polynomial time, it is not currently known if unconditional superlinear time complexity lower bounds exist for log-CircuitSAT. We prove a superlinear conditional lower bound for log-CircuitSAT, as our main contribution. (Our conditional lower bound is in fact superquasilinear, where f is a quasilinear function if $f(n)/n \in \text{polylog}(n)$.) In particular, we show in Theorem 22 that log-CircuitSAT is not decidable in essentially-linear time unless the Exponential Time Hypothesis (ETH) is false. This result is significant because existing works have only obtained conditional lower bounds for polynomial time problems based on the Strong Exponential Time Hypothesis (SETH). It is well known that SETH implies ETH but the reverse implication is not known to hold [16, Theorem 14.5]. In fact, it has been claimed that while ETH is plausible, SETH "is regarded by many as a quite doubtful working hypothesis that can be refuted at any time" [16, p. 470]. We therefore believe that a conditional lower bound for a natural polynomial time problem based on ETH instead of SETH represents significant progress.

As a further contribution, in Theorem 24 we show that log-CircuitSAT is not decidable in essentially-linear time unless k-Clique is decidable in essentially-linear time in terms of the graph's size for all fixed k. This result is significant because the current best known algorithm for deciding k-Clique runs in $O(v^{0.792k})$ time [40] where v denotes the number of vertices. Furthermore, showing that there exists a constant c such that k-Clique is decidable in $O(v^c)$ time for all fixed k would constitute a major breakthrough.

Our results for log-CircuitSAT follow from Speed-up Theorems 12, 14, and 16. These theorems show how a small runtime improvement for the deterministic simulation of nondeterministic machines with short witnesses would imply a substantial runtime improvement for the deterministic simulation of nondeterministic machines with large witnesses. Furthermore, these results advance our knowledge of limited nondeterminism by exploring possible trade-offs between time and witness length.

Our techniques are straightforward adaptations of existing approaches to simulation; our contribution is a more detailed analysis of these simulations and how they behave when composed and iterated. This more detailed analysis is made possible by our novel approach to limited nondeterminism in Section 3.1.

2 Background

Let $\mathbb{N}$ denote the set of positive integers $\{1, 2, \dots\}$. The class of polynomial functions is

$$\operatorname{poly}(n) = \bigcup_{c>0} \{\ f(n) \colon \mathbb{N} \to \mathbb{N} \ \mid \ f(n) = O(n^c)\ \},$$

and in a slight abuse of notation, we also sometimes use $\operatorname{poly}(n)$ to mean an arbitrary function from this class. We also refer to the class of polylogarithmic functions

$$\operatorname{polylog}(n) = \bigcup_{c>0} \{\ f(n) \colon \mathbb{N} \to \mathbb{N} \ \mid \ f(n) = O((\log(n))^c)\ \}.$$

2.1 Conditional Lower Bounds

Fine-grained complexity theory is a subject focused on exact runtime bounds and conditional lower bounds. A conditional lower bound for a polynomial time problem typically takes the following form: polynomial time problem A is not decidable in $O(n^{\alpha-\varepsilon})$ time for all $\varepsilon > 0$ assuming that problem B is not decidable in $O(t(n)^{\beta-\varepsilon})$ time for all $\varepsilon > 0$, where α and β are constants and $t(n)$ is a function (typically $t(n)$ is either polynomial or exponential). This is referred to as a conditional lower bound because problem A has a lower bound under the assumption that B has a lower bound. Conditional lower bounds are known for many polynomial time problems including Triangle Finding, Orthogonal Vectors Problem (OVP), 3SUM, and All Pairs Shortest Path (APSP) [45, 7].

2.2 Exponential Time Hypothesis

Decision problems related to Boolean formulas have been significant to the study of computational hardness [13, 32]. As a result, satisfiability of Boolean formulas (SAT) is a natural candidate for lower bound assumptions. In particular, it is common to focus on satisfiability of Boolean formulas in conjunctive normal form with clause width at most k (denoted by k-CNF-SAT) for a fixed k.

The exponential time hypothesis (ETH) states that there is some $\varepsilon > 0$, such that 3-CNF-SAT cannot be decided in $\operatorname{poly}(n) \cdot 2^{\varepsilon \cdot v}$ time, where n denotes the input size and v denotes the number of variables [26]. The strong exponential time hypothesis (SETH) states that for every $\varepsilon > 0$, there is a sufficiently large k such that k-CNF-SAT cannot be decided in $\operatorname{poly}(n) \cdot 2^{(1-\varepsilon)\cdot v}$ time [26, 27, 11].

Conditional lower bounds are frequently shown relative to k-CNF-SAT. For instance, it is well known that the Orthogonal Vectors Problem (OVP) on polylogarithmic length vectors is not decidable in $O(n^{2-\varepsilon})$ time for all $\varepsilon > 0$ assuming SETH [42, 45, 7].

▶ Remark 1. As far as we know, the current best reduction shows that an $O(n^{\alpha})$ time algorithm for OVP would lead to a $\operatorname{poly}(n) \cdot 2^{\frac{\alpha \cdot v}{2}}$ time algorithm for k-CNF-SAT for all k [42, 45, 7]. This isn't sufficient to show a lower bound conditional on ETH. Furthermore, we do not know if the existence of an essentially-linear time algorithm for OVP would imply that ETH is false.

2.3 Limited Nondeterminism

A nondeterministic polynomial time problem is a problem that can be decided in polynomial time by a nondeterministic machine. Nondeterminism can appear in a computation in multiple different ways. For instance, a machine could have nondeterministic bits written on a tape in advance, or could make nondeterministic guesses during the computation. Such variations do not appear to make much difference for nondeterministic polynomial time (NP). However, the definitions do require special care for notions of limited nondeterminism, which refers to the restriction or bounding of the amount of nondeterminism in a computation.

We proceed by reviewing some prior models of limited nondeterminism. Consider a machine model consisting of a multitape Turing machine with a special guess tape; all the remaining tapes are standard. In this model, the number of bits of nondeterminism used by the machine is the number of cells of the guess tape that are accessed by the machine during a computation, multiplied by the number of bits represented by each cell. The contents of the guess tape are referred to as the *witness.*

Kintala and Fischer [29, 30] defined $\mathsf{P}_{f(n)}$ as the class of languages that can be decided by a polynomial time bounded machine which scans at most $f(n)$ cells of the guess tape for each input of size n. Note that this concept uses an exact limit for the amount of nondeterminism. Abandoning this exactness, Àlvarez, Díaz and Torán [4, 17], making explicit a concept of Xu, Doner, and Book [35], then defined β_k as the class of languages that can be decided by a polynomial-time bounded machine which uses at most $O((\log(n))^k)$ bits of nondeterminism. Farr took a similar approach [18], defining $f(n)$-NP as the languages that can be decided by a polynomial-time bounded machine which scans at most $f(q(n))$ cells of the guess tape for each input of size n. Here q is a polynomial that depends only on the machine, so again there is an unspecified constant factor allowed in the amount of nondeterminism. Note that $f(n)$-NP is the union over all k of the classes $\mathsf{P}_{f(n^k)}$. Another related approach was taken by Buss and Goldsmith [8] where $\mathsf{N}^m\mathsf{P}_l$ is defined as the class of languages decided by nondeterministic machines in quasi-n^l time making at most $m \cdot \log(n)$ nondeterministic guesses. In this approach the limit on the amount of nondeterminism is exact, but arbitrary poly-logarithmic factors are allowed in the time bound. Finally, in the survey by Goldsmith, Levy and Mundhenk [21] the β_k classes were then extended to verifiers other than those with a polynomial time bound. In this notation, $\beta_k{-}\mathsf{C}$ is defined relative to a complexity class C that bounds the power of the verifier. Therefore, we have $\beta_k = \beta_k{-}\mathsf{P}$.

Taking a slightly different approach, Cai and Chen [10] focused on machines that partition access to nondeterminism, by first creating the contents of the guess tape, and then using a deterministic machine to check this guess. In this terminology, $\mathsf{GC}(s(n), \mathsf{C})$ is the class of languages that can be decided by a machine that guesses $O(s(n))$ bits and then uses the power of class C to verify. Again an arbitrary constant factor is allowed in the number of nondeterministic bits, to allow classes to contain complete languages.

Santhanam [38] then returned to a definition that uses an exact limit for the amount of allowed nondeterminism: $\mathsf{NTIGU}(t(n), g(n))$ is the class of languages that can be decided by a machine that makes $g(n)$ guesses and runs for $O(t(n))$ time. These classes have also been denoted $\mathsf{NTIMEGUESS}(t(n), g(n))$ in a more recent work [20].

It follows from the definitions that

$$\mathsf{P} = \mathsf{P}_{O(\log(n))} = \mathsf{NTIGU}(\text{poly}(n), O(\log(n))) = \mathsf{GC}(O(1), \mathsf{P}) = \mathsf{N}^{O(1)}\mathsf{P}_{O(1)} = \beta_1$$

and

$$\mathsf{NP} = n\text{-}\mathsf{NP} = \mathsf{P}_{n^{O(1)}} = \mathsf{NTIGU}(\text{poly}(n), \text{poly}(n)) = \mathsf{GC}(n^{O(1)}, \mathsf{P}).$$

Furthermore, the β_k classes are meant to capture classes between P and NP.

▶ Remark 2. In this work, we focus on the log-CircuitSAT problem and the levels within $\mathsf{P} = \beta_1$. It is worth noting that a loosely related work [19] investigated the log-Clique problem which is in $\beta_2 = \mathsf{NTIGU}(\text{poly}(n), O(\log(n)^2))$.

3 Time-Witness Trade-offs

In the following, we introduce a new notion of limited nondeterminism that we use to prove new relationships between deterministic and nondeterministic computations. In particular, we prove that if faster deterministic algorithms exist, then there are straightforward trade-offs between time and witness length. For our notion of limited nondeterminism, unlike existing models, the nondeterministic guesses are preallocated as placeholders within an input string. These placeholders can then be filled with nondeterministic bits. It is important to note that different models of limited nondeterminism could be used. However, our model allows us to preserve the input size enabling us to prove technical results, Lemmas 9 and 10. Attempting to prove a result like Lemma 9 for a model such as NTIGU introduces unnecessary challenges with managing input and guess strings.

3.1 A New Model for Limited Nondeterminism

The attempts at constructing robust classes containing complete problems by allowing arbitrary factors in the amount of nondeterminism were challenged by the various downward collapses of the β hierarchy shown by Beigel and Goldsmith relative to oracles [5]. We therefore need a notion of limited nondeterminism that tracks constant factors in the amount of nondeterminism, accepting a lack of complete problems in our complexity classes to gain greater precision in reductions. This suggests using the NTIGU notation.

However, we found that attempting to use the NTIGU notion directly leads to difficulties with bookkeeping when composing multiple reductions because of the necessary simultaneous management of input and guess strings. Since composing reductions is at the heart of our approach for proving speed-up theorems in Subsection 3.3, we sought a different notion that overcomes these unnecessary technical obstacles.

We now introduce our model of limited nondeterminism which allows us to be more explicit than the NTIGU classes in keeping track of the witness bits when composing reductions. Reminiscent of the Cai and Chen guess-and-check classes, in our model the nondeterministic guesses will be preallocated as placeholder characters within an input string. This means that we can only fill in placeholder characters with nondeterministic bits. This property is essential for proving structural properties (see the translation and padding lemmas in Subsection 3.2). With other models, proofs of structural properties appear to be intrinsically more complex, requiring separate treatment of various overheads and applications of tape reduction theorems.

Consider strings over a ternary alphabet $\Sigma = \{0, 1, p\}$ where p is referred to as the placeholder character. We index the bits of a string starting from position 0. For any string $x \in \Sigma^*$, we let $|x|$ denote the length of x and $\#_p(x)$ denote the number of placeholder character occurrences in x.

▶ **Definition 3.** *Let a string $r \in \{0,1\}^*$ be given. Define a function*

$$sub_r : \Sigma^* \to \Sigma^*$$

such that for each string $x \in \Sigma^*$, $sub_r(x)$ is obtained from x by replacing placeholder characters with bits from r so that the ith placeholder character from x is replaced by the ith bit of r for all i satisfying $0 \leq i < min\{|r|, \#_p(x)\}$. Also, define $SUB(n) := \{\ sub_r \mid |r| \leq n\ \}$. We call sub_r a prefix filling and $SUB(n)$ a set of prefix fillings.

▶ **Example 4.** Consider strings $x = 11p01p0p$ and $r = 0110$. By applying the preceding definition, we have that $sub_r(x) = 11001101$.

A prefix filling sub_r replaces the first $|r|$ placeholder characters with the bits of r in order (from the least index to the greatest). If there are fewer placeholder characters then some of the bits of r remain unused. We also consider the notion of an *unrestricted filling*, which is any injective replacement of placeholder characters, without specifying the particular order.

▶ **Definition 5.** *For strings x and $y \in \Sigma^*$, we write $x \preceq y$ if x can be obtained from y by replacing any number of placeholder characters in y with 0 or 1. Given a language $L \subseteq \Sigma^*$, we let the closure of L under unrestricted fillings be*

$$Clo(L) := \{\, x \in \Sigma^* \mid (\exists\, y \in L)\; x \preceq y \,\}.$$

▶ **Example 6.** Consider a language $L = \{0p1p\}$. By applying the preceding definition, we have that $\mathrm{Clo}(L) = \{0p1p, 0p10, 0p11, 001p, 011p, 0010, 0011, 0110, 0111\}$.

In the following, $\mathsf{DTIME}(t(n))$ represents the class of languages decidable in $O(t(n))$ time by multitape Turing machines that have read and write access to all tapes (including the input tape). We proceed by defining a complexity class $\mathsf{DTIWI}(t(n), w(n))$ where intuitively $t(n)$ represents a time bound and $w(n)$ represents a bound on witness length.

▶ **Definition 7.** *Let $\Sigma = \{0, 1, p\}$ and consider a language $L \subseteq \Sigma^*$. We write*

$$L \in \mathsf{DTIWI}(t(n), w(n))$$

if there exist languages U and $V \subseteq \Sigma^*$ satisfying the following properties:

- *$U \in \mathsf{DTIME}(n)$,*
- *$Clo(U) \in \mathsf{DTIME}(n)$,*
- *$V \in \mathsf{DTIME}(t(n))$, and*
- *for all $x \in \Sigma^*$, $x \in L$ if and only if $x \in U$ and there exists $s \in SUB(w(|x|))$ such that $s(x) \in V$.*

We refer to V as a verification language for L with input string universe U.

There are many different ways to encode structures as strings over a fixed alphabet, so decision problems can take many different forms as formal languages. To put a problem within $\mathsf{DTIWI}(t(n), w(n))$, we therefore need to provide an encoding for its inputs with placeholder characters at the appropriate positions.

▶ **Example 8.** SAT can be represented such that each input has placeholder characters out front followed by an encoding of a Boolean formula. Each variable is represented as a binary number representing an index to a placeholder. The placeholders will be nondeterministically filled to create a variable assignment.

3.2 Structural Properties of Limited Nondeterminism

The following two lemmas demonstrate structural properties relating time and witness length. These properties will be essential to proving speed-up theorems in Subsection 3.3 that reveal new relationships between deterministic and nondeterministic computations.

▶ **Lemma 9** (Translation Lemma). *If $\mathsf{DTIWI}(t(n), w(n)) \subseteq \mathsf{DTIME}(t'(n))$, then for all w',*

$$\mathsf{DTIWI}(t(n), w(n) + w'(n)) \subseteq \mathsf{DTIWI}(t'(n), w'(n)).$$

Proof. Suppose that $\mathsf{DTIWI}(t(n), w(n)) \subseteq \mathsf{DTIME}(t'(n))$.

Let a function w' be given. Let $L \in \mathsf{DTIWI}(t(n), w(n) + w'(n))$ be given. By definition, there exist an input string universe U and a verification language $V \in \mathsf{DTIME}(t(n))$ satisfying that $\forall x \in \Sigma^*$, $x \in L$ if and only if $x \in U$ and there exists $s \in \mathrm{SUB}(w(|x|) + w'(|x|))$ such that $s(x) \in V$. Consider a new language

$$L' := \{\, x \in \mathrm{Clo}(U) \mid (\exists s \in \mathrm{SUB}(w(|x|)))\ s(x) \in V \,\}.$$

By interpreting V as a verification language for L' with input string universe $\mathrm{Clo}(U)$, we get $L' \in \mathsf{DTIWI}(t(n), w(n))$. By assumption, it follows that

$$L' \in \mathsf{DTIME}(t'(n)).$$

Finally, by interpreting L' as a verification language for L with input string universe U, we get $L \in \mathsf{DTIWI}(t'(n), w'(n))$. ◀

Recall that a function $f : \mathbb{N} \to \mathbb{N}$ is *fully time-constructible* if there is a deterministic multitape Turing machine M that for every input of length n runs for exactly $f(n)$ steps [25]. By convention, if $f(n)$ is a fully time-constructible function, then $f(n) \geq n$ for all $n \in \mathbb{N}$.

▶ **Lemma 10** (Padding Lemma). *If* $\mathsf{DTIWI}(t(n), w(n)) \subseteq \mathsf{DTIME}(t'(n))$, *then for all fully time-constructible functions* f*,*

$$\mathsf{DTIWI}(t(f(n)), w(f(n))) \subseteq \mathsf{DTIME}(t'(f(n))).$$

Proof. Suppose that $\mathsf{DTIWI}(t(n), w(n)) \subseteq \mathsf{DTIME}(t'(n))$, and that $f : \mathbb{N} \to \mathbb{N}$ is fully time-constructible. By definition, there exist an input string universe U and a verification language

$$V \in \mathsf{DTIME}(t(f(n)))$$

so that $\forall x \in \Sigma^*$, $x \in L$ if and only if $x \in U$ and there exists

$$s \in \mathrm{SUB}(w(f(|x|)))$$

such that $s(x) \in V$. Consider new languages L', V', and U' such that

$$\begin{aligned} L' &:= \{\, 1^{k-1} \cdot 0 \cdot x \mid k + |x| = f(|x|) \ \wedge\ x \in L \,\}, \\ V' &:= \{\, 1^{k-1} \cdot 0 \cdot x \mid k + |x| = f(|x|) \ \wedge\ x \in V \,\}, \text{ and} \\ U' &:= \{\, 1^{k-1} \cdot 0 \cdot x \mid k \geq 1 \ \wedge\ x \in U \,\}. \end{aligned}$$

Since $V \in \mathsf{DTIME}(t(f(n)))$, we have that $V' \in \mathsf{DTIME}(t(n))$. By interpreting V' as a verification language for L' with input string universe U', we get $L' \in \mathsf{DTIWI}(t(n), w(n))$. By assumption, it follows that $L' \in \mathsf{DTIME}(t'(n))$. We conclude that $L \in \mathsf{DTIME}(t'(f(n)))$. ◀

▶ Remark 11. Initially, we tried to use other notions of limited nondeterminism such as NTIGU to prove the preceding lemmas. However, the proofs were messy and required increasing the number of Turing machine tapes or the time complexity. In contrast, our model for limited nondeterminism (DTIWI) preserves the input size leading to straightforward proofs with tighter complexity bounds.

3.3 Speed-up Theorems

In this subsection, we carefully prove three speed-up theorems relating time and witness length. It is important to mention that there are existing speed-up theorems in the recent literature relating different computational resources such as those relating time and space in [43, 9] and relating probabilistic circuit size and success probability in [36]. In addition, although relevant, we note that our speed-up results are distinct from recent hardness magnification results [12] which amplify circuit lower bounds rather than speed up computations.

The first speed-up theorem follows by repeatedly applying the structural properties of limited nondeterminism from the preceding subsection.

▶ **Theorem 12** (First Speed-up Theorem)**.** *Let α be a rational number such that $1 \leq \alpha < 2$. If*

$$\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha}),$$

then for all $k \in \mathbb{N}$, $\mathsf{DTIWI}(n, (\Sigma_{i=0}^{k} \alpha^i) \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+1}})$.

Proof. Suppose α is rational and $1 \leq \alpha < 2$. (Note that when $\alpha = 1$ some of the following formulas can be simplified, but the proof still holds for this case.)

Now suppose that $\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha})$. We prove by induction on k that for all $k \in \mathbb{N}$,

$$\mathsf{DTIWI}(n, (\Sigma_{i=0}^{k} \alpha^i) \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+1}}).$$

The base case ($k = 0$) is true by assumption. For the induction step, suppose that

$$\mathsf{DTIWI}(n, (\Sigma_{i=0}^{k} \alpha^i) \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+1}}).$$

By applying this assumption with Lemma 9, we get that

$$\mathsf{DTIWI}(n, (\Sigma_{i=0}^{k+1} \alpha^i) \log(n)) \subseteq \mathsf{DTIWI}(n^{\alpha^{k+1}}, \alpha^{k+1} \cdot \log(n)).$$

Let $f(n) = n^{\alpha^{k+1}}$, which is fully time-constructible [31, Example 1]. Next, we apply our initial assumption and Lemma 10 with $f(n)$, $w(n) = \log(n)$, $t(n) = n$, and $t'(n) = n^{\alpha}$. Therefore,

$$\mathsf{DTIWI}(n^{\alpha^{k+1}}, \alpha^{k+1} \cdot \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+2}}).$$

It follows that $\mathsf{DTIWI}(n, (\Sigma_{i=0}^{k+1} \alpha^i) \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+2}})$. ◀

▶ Remark 13. Theorem 12 is a speed-up result because when $1 \leq \alpha < 2$, the exponent from the runtime divided by the constant factor for the witness string length decreases as k increases. In particular, we have

$$\lim_{k \to \infty} \frac{\alpha^{k+1}}{\Sigma_{i=0}^{k} \alpha^i} = (\alpha - 1) \cdot \lim_{k \to \infty} \frac{\alpha^{k+1}}{\alpha^{k+1} - 1} = \alpha - 1 < 1.$$

The second speed-up theorem follows by combining the first speed-up theorem with the padding lemma. We say that a function $g \colon \mathbb{N} \to \mathbb{N}$ is *well-computable* if $g(n) \leq n$ for every $n \in \mathbb{N}$, $g(n) = \omega(\log(n))$, and $g(n)$ can be computed in $\mathrm{poly}(n)$ steps.

▶ **Theorem 14** (Second Speed-up Theorem)**.** *Suppose that g is a well-computable function. Let α be a rational number such that $1 < \alpha < 2$. If*

$$\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha}),$$

then

$$(\forall \varepsilon > 0)\ \mathsf{DTIWI}(\mathrm{poly}(n), g(n)) \subseteq \mathsf{DTIME}(2^{(1+\varepsilon) \cdot (\alpha - 1) \cdot g(n)}).$$

Proof. Let α be a rational number such that $1 < \alpha < 2$. Let $z(\alpha, k) = \Sigma_{i=0}^{k} \alpha^i$. Note that

$$z(\alpha, k) = \frac{\alpha^{k+1} - 1}{\alpha - 1}.$$

Suppose that $\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha})$. Let $\varepsilon > 0$ be given. By Theorem 12, we have that for all $k \in \mathbb{N}$,

$$\mathsf{DTIWI}(n, z(\alpha, k) \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+1}}).$$

Let $f(n) = 2^{\lceil g(n)/z(\alpha,k) \rceil}$ if $g(n) \geq z(\alpha, k) \log(n)$ and $f(n) = n$ otherwise. Because $g(n) = \omega(\log(n))$, there is some $c > 1$ such that $f(n) > cn$ for all but finitely many $n \in \mathbb{N}$. Now, since $g(n)$ can be computed in $\text{poly}(n)$ time and $z(\alpha, k)$ is rational, $f(n)$ can be computed in binary in $\text{poly}(n)$ time. Furthermore, since $f(n)$ is superpolynomial, $f(n)$ can be computed in $O(f(n))$ time. Therefore, by [31, Theorem 4.1], $f(n)$ is fully time-constructible. Next, we apply Lemma 10 with $f(n)$, $w(n) = z(\alpha, k) \log(n)$, and $t(n) = n$. Therefore

$$\mathsf{DTIWI}(2^{g(n)/z(\alpha,k)}, g(n)) \subseteq \mathsf{DTIME}(2^{(\alpha^{k+1}) \cdot g(n)/z(\alpha,k)}).$$

Again, since $2^{g(n)/z(\alpha,k)}$ is superpolynomial, we have

$$\mathsf{DTIWI}(\text{poly}(n), g(n)) \subseteq \mathsf{DTIME}(2^{(\alpha^{k+1}) \cdot g(n)/z(\alpha,k)}).$$

Then, since

$$\lim_{k \to \infty} \frac{\alpha^{k+1}}{\alpha^{k+1} - 1} = 1,$$

there exists k sufficiently large such that

$$\frac{\alpha^{k+1}}{\alpha^{k+1} - 1} \leq 1 + \varepsilon.$$

Therefore, by choosing sufficiently large k, we have

$$\mathsf{DTIWI}(\text{poly}(n), g(n)) \subseteq \mathsf{DTIME}(2^{(1+\varepsilon) \cdot (\alpha - 1) \cdot g(n)}).$$ ◀

▶ **Corollary 15.** *Suppose that g is a well-computable function. If for all $\alpha > 1$,*

$$\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha}),$$

then $(\forall \varepsilon > 0)$ $\mathsf{DTIWI}(\text{poly}(n), g(n)) \subseteq \mathsf{DTIME}(2^{\varepsilon \cdot g(n)})$.

Proof. Follows directly from Theorem 14. ◀

The third speed-up theorem follows by carefully applying the first speed-up theorem.

▶ **Theorem 16** (Third Speed-up Theorem). *If for all $\alpha > 1$,*

$$\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha}),$$

then for all $k \in \mathbb{N}$ and all $\alpha > 1$, $\mathsf{DTIWI}(n, k \cdot \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha})$.

Proof. Suppose that for all $\alpha > 1$, $\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha})$. By Theorem 12, for all rational α such that $1 < \alpha < 2$ and for all $k \in \mathbb{N}$,

$$\mathsf{DTIWI}(n, (\Sigma_{i=0}^{k} \alpha^{i}) \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+1}}).$$

Notice that when $\alpha > 1$, we have $k < (\Sigma_{i=0}^{k} \alpha^{i})$. Therefore, for all rational α such that $1 < \alpha < 2$ and for all $k \in \mathbb{N}$,

$$\mathsf{DTIWI}(n, k \cdot \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha^{k+1}}).$$

Now, let $k \in \mathbb{N}$ and a rational number $\alpha_1 > 1$ be given. We can choose a rational number $\alpha_2 > 1$ sufficiently close to 1 so that $\alpha_2^{k+1} \leq \alpha_1$. It follows that

$$\mathsf{DTIWI}(n, k \cdot \log(n)) \subseteq \mathsf{DTIME}(n^{\alpha_2^{k+1}}) \subseteq \mathsf{DTIME}(n^{\alpha_1}).$$

As the rationals form a dense subset of the reals, the result follows. ◀

4 Superlinear Conditional Lower Bounds

4.1 log-CircuitSAT Decision Problem

A common generalization of SAT is the problem of deciding satisfiability of Boolean circuits (denoted by CircuitSAT). There is a natural restriction of CircuitSAT to bounded fan-in Boolean circuits with m gates and $\log(m)$ inputs (denoted by log-CircuitSAT) [8, 2]. We encode this problem so that the placeholder characters are out front followed by an encoding of a bounded fan-in Boolean circuit. Such an encoding can be carried out so that if n denotes the total input length and m denotes the number of gates, then $n = \Theta(m \cdot \log(m))$.

The log-CircuitSAT decision problem is decidable in polynomial time because we can evaluate the circuit on every possible input assignment. Whether or not we can decide log-CircuitSAT in $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$ is an open problem. Furthermore, as far as we know, no unconditional superlinear lower bounds are known for log-CircuitSAT. Later in this section, we prove a superlinear conditional lower bound for log-CircuitSAT. In particular, we show that if log-CircuitSAT is decidable in essentially-linear time, then ETH is false (Theorem 22) meaning that a small runtime improvement for log-CircuitSAT would lead to a substantial runtime improvement for NP-complete problems.

4.2 Simulating Turing Machines Using Boolean Circuits

Let a fully time-constructible function t be given. Any $O(t(n))$ time bounded Turing machine can be simulated by an oblivious Turing machine in $O(t(n) \cdot \log(t(n)))$ time [37]. Moreover, any $O(t(n))$ time bounded Turing machine can be simulated by Boolean circuits of size $O(t(n) \cdot \log(t(n)))$ which can be computed efficiently by a Turing machine [14].

▶ **Theorem 17** ([37, 14, 8, 28]). *Let a fully time-constructible function t be given. If $L \in \mathsf{DTIME}(t(n))$, then in*

$$O(t(n) \cdot \mathrm{poly}(\log(t(n))))$$

time, we can compute Boolean circuits for L of size at most $O(t(n) \cdot \log(t(n)))$.

We now use Theorem 17 to show that any problem in $\mathsf{DTIWI}(n, \log(n))$ is efficiently reducible to log-CircuitSAT.

▶ **Theorem 18.** *Any $L \in \mathsf{DTIWI}(n, \log(n))$ is reducible to logarithmically many instances of* log-CircuitSAT *in essentially-linear time by a Turing machine.*

Proof. Let $L \in \mathsf{DTIWI}(n, \log(n))$ be given. Let $V \in \mathsf{DTIME}(n)$ denote a verification language for L with input string universe $U \in \mathsf{DTIME}(n)$.

Let an input string $x \in U$ of length n be given. By Theorem 17, we can compute a Boolean circuit[1] C for V with at most $O(n \cdot \log(n))$ gates in essentially-linear time on a Turing machine. In the following, let $[\log(n)]$ denote $\{1, 2, \ldots, \lfloor \log(n) \rfloor\}$. Now, we construct a family of circuits $\{C_i\}_{[\log(n)]}$ such that for each $i \in [\log(n)]$, C_i is obtained by fixing the characters of x into the circuit C so that only i input bits remain where these input bits are associated with the first i placeholders within x. Therefore the circuit C_i has at most $\log(n)$ inputs and at most $O(n \cdot \log(n))$ gates. It follows that $x \in L$ if and only if there exists $i \in [\log(n)]$ such that C_i is satisfiable. ◀

▶ **Corollary 19.** *If for all $\alpha > 1$ we have* log-CircuitSAT $\in \mathsf{DTIME}(n^\alpha)$*, then for all $\alpha > 1$*

$$\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n^\alpha).$$

Proof. Follows directly from Theorem 18. ◀

4.3 ETH-hardness

We combine results from Subsection 4.2 with the Second Speed-up Theorem to prove superlinear conditional lower bounds for log-CircuitSAT. In particular, existence of essentially-linear time algorithms for log-CircuitSAT would imply that ETH is false.

▶ **Corollary 20.** *Suppose that g is a well-computable function. If for all $\alpha > 1$,*

$$\text{log-CircuitSAT} \in \mathsf{DTIME}(n^\alpha),$$

then $(\forall \varepsilon > 0)\ \mathsf{DTIWI}(\mathrm{poly}(n), g(n)) \subseteq \mathsf{DTIME}(2^{\varepsilon \cdot g(n)})$.

Proof. Follows by combining Corollary 19 with Corollary 15. ◀

We now relate log-CircuitSAT and CircuitSAT, showing that an essentially-linear upper bound for log-CircuitSAT would imply a subexponential upper bound for CircuitSAT.

▶ **Theorem 21.** *If for every $\alpha > 1$ we have that* log-CircuitSAT $\in \mathsf{DTIME}(n^\alpha)$*, then*

$$(\forall \varepsilon > 0)\ \text{CircuitSAT} \in \mathsf{DTIME}(\mathrm{poly}(n) \cdot 2^{\varepsilon \cdot m}),$$

where m is the number of gates.

Proof. Suppose that for all $\alpha > 1$, log-CircuitSAT $\in \mathsf{DTIME}(n^\alpha)$. Letting $\lg x = \max\{1, \log(x)\}$ and applying Corollary 20 with $g(n) = \frac{n}{\lg(n)}$ (which is well-computable), we conclude that

$$(\forall \varepsilon > 0)\ \mathsf{DTIWI}(\mathrm{poly}(n), \frac{n}{\log(n)}) \subseteq \mathsf{DTIME}(2^{\varepsilon \cdot \frac{n}{\log(n)}}).$$

[1] Since L and V are over a ternary alphabet, the input strings are encoded into binary before being fed into the Boolean circuits.

Recall that we encode Boolean circuits so that $n = \Theta(m \cdot \log(m))$ where m is the number of gates. Therefore, $\frac{n}{\log(n)}$ is $\Theta(m)$. Also, under reasonable encoding conventions, $\frac{n}{\log(n)}$ will actually be larger than the number of gates and inputs. (Note that one could always scale up the witness size by a rational constant factor if needed.) Hence,

$$\text{CircuitSAT} \in \mathsf{DTIWI}(\text{poly}(n), \frac{n}{\log(n)}).$$

Therefore, $(\forall \varepsilon > 0)$ CircuitSAT $\in \mathsf{DTIME}(2^{\varepsilon \cdot \frac{n}{\log(n)}})$. It follows that

$$(\forall \varepsilon > 0) \text{ CircuitSAT} \in \mathsf{DTIME}(\text{poly}(n) \cdot 2^{\varepsilon \cdot m}).$$ ◀

We now show that log-CircuitSAT cannot be decided in essentially-linear time unless ETH fails. Note that this is a conditional lower bound based on ETH rather than the more common (and stronger) SETH assumption.

▶ **Theorem 22.** *If* log-CircuitSAT $\in \mathsf{DTIME}(n^\alpha)$ *for every* $\alpha > 1$*, then* ETH *is false.*

Proof. Because 3-CNF-SAT is a special case of CircuitSAT, Theorem 21 implies that

$$(\forall \varepsilon > 0) \text{ 3-CNF-SAT} \in \mathsf{DTIME}(\text{poly}(n) \cdot 2^{\varepsilon \cdot m})$$

where m is the number of bounded AND, OR, and NOT gates (which is approximately three times the number of clauses). By applying the Sparsification Lemma [27, 33], we get that

$$(\forall \varepsilon > 0) \text{ 3-CNF-SAT} \in \mathsf{DTIME}(\text{poly}(n) \cdot 2^{\varepsilon \cdot v})$$

where v is the number of variables. It follows that ETH is false. ◀

4.4 Hardness for k-Clique

We combine results from Subsection 4.2 with the Third Speed-up Theorem to prove that the existence of essentially-linear time algorithms for log-CircuitSAT would imply that k-Clique has essentially-linear time algorithms for all fixed k. It is important to note that our construction is non-uniform meaning that we obtain differing algorithms that cannot necessarily be combined into a single efficient approach for solving k-Clique on non-constant k. As a result, our argument is not sufficient to conclude $\mathsf{FPT} = \mathsf{W}[1]$.

▶ **Corollary 23.** *If for every* $\alpha > 1$ *we have that* log-CircuitSAT $\in \mathsf{DTIME}(n^\alpha)$*, then*

$$\mathsf{DTIWI}(n, k \cdot \log(n)) \subseteq \mathsf{DTIME}(n^\alpha)$$

for every $k \in \mathbb{N}$ *and every* $\alpha > 1$*.*

Proof. Follows by combining Corollary 19 with Theorem 16. ◀

From this, we are able to obtain a meaningful connection between the log-CircuitSAT and k-Clique problems.

▶ **Theorem 24.** *If for every* $\alpha > 1$ *we have that* log-CircuitSAT $\in \mathsf{DTIME}(n^\alpha)$*, then*

$$k\text{-Clique} \in \mathsf{DTIME}(n^\alpha)$$

for every $k \in \mathbb{N}$ *and every* $\alpha > 1$*.*

Proof. The variable n denotes the total length of the graph's encoding which is $\Theta((v+e) \cdot \log(v))$ where v is the number of vertices and e is the number of edges. We observe that for all fixed k, we have

$$k\text{-Clique} \in \mathsf{DTIWI}(n, k \cdot \log(n)).$$

We combine this observation with Corollary 23 to obtain the desired result. ◀

▶ Remark 25. Although we do not focus on parameterized complexity theory here, the preceding arguments can also be used to show that if log-CircuitSAT is decidable in essentially-linear time, then $\mathsf{W}[1] \subseteq$ non-uniform-FPT. Moreover, we suggest that this implication could be extended to $\mathsf{W}[\mathsf{P}] \subseteq$ non-uniform-FPT. We refer the reader to [2] for background on $\mathsf{W}[\mathsf{P}]$ and the W hierarchy.

5 Conclusion

We have demonstrated superlinear conditional lower bounds for the log-CircuitSAT decision problem by carefully investigating properties of limited nondeterminism. In particular, in Theorem 22 we showed that the existence of essentially-linear time Turing machines for log-CircuitSAT would imply that ETH is false. This means that a small runtime improvement for log-CircuitSAT would lead to a substantial runtime improvement for NP-complete problems. Through this investigation we revealed new relationships between deterministic and nondeterministic computations.

We leave two important questions unanswered that we hope will inspire future work.

▶ **Question 26.** *Would the existence of essentially-linear time random access machines for* log-CircuitSAT *imply that* ETH *is false? This question is related to whether linear time for random access machines can be simulated in subquadratic time by multitape Turing machines [15]. It is also related to whether random access machines can be made oblivious [22].*

▶ **Question 27.** *Can the construction from the first speed-up theorem (Theorem 12) be carried out for a non-constant number k of iterations? We speculate that if it can, then* $\mathsf{DTIWI}(n, \log(n)) \subseteq \mathsf{DTIME}(n \cdot \log(n))$ *would imply that* $\mathsf{NTIME}(n) \subseteq \mathsf{DTIME}(2^{\sqrt{n}})$.

In addition, although this work does not focus on circuit lower bounds, we suggest that recent results connecting the existence of faster algorithms with circuit lower bounds [1, 44, 43, 6] could be applied to show that the existence of faster algorithms for log-CircuitSAT would imply new circuit lower bounds for E^{NP} as well as other complexity classes.

Finally, we leave the reader with the thought that the speed-up theorems for limited nondeterminism (Theorems 12, 14, and 16) might be special cases of a more general speed-up result connecting nondeterminism, alternation, and time.

References

1 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: Or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, STOC 2016, pages 375–388. Association for Computing Machinery, 2016. doi:10.1145/2897518.2897653.

2 Karl A. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995. doi:10.1016/0168-0072(94)00034-Z.

3 Akeo Adachi, Shigeki Iwata, and Takumi Kasai. Some combinatorial game problems require $\Omega(n^k)$ time. *J. ACM*, 31(2):361–376, March 1984. doi:10.1145/62.322433.

4 Carme Àlvarez, Josep Díaz, and Jacobo Torán. Complexity classes with complete problems between P and NP-C. In J. Csirik, J. Demetrovics, and F. Gécseg, editors, *FCT 1989: Proceedings of the 7th International Conference on Fundamentals of Computation Theory*, volume 380 of *LNCS*, pages 13–24. Springer, 1989. doi:10.1007/3-540-51498-8_2.

5 R. Beigel and J. Goldsmith. Downward separation fails catastrophically for limited nondeterminism classes. *SIAM Journal on Computing*, 27(5):1420–1429, 1998. doi:10.1137/S0097539794277421.

6 Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014: International Colloquium on Automata, Languages, and Programming*, volume 8572 of *LNCS*, pages 163–173. Springer, 2014. doi:10.1007/978-3-662-43948-7_14.

7 Karl Bringmann. Fine-grained complexity theory (tutorial). In Rolf Niedermeier and Christophe Paul, editors, *STACS 2019: 36th International Symposium on Theoretical Aspects of Computer Science*, volume 126 of *LIPIcs*, pages 4:1–4:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.4.

8 Jonathan Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.

9 Jonathan Buss and Kenneth Regan. Simultaneous bounds on time and space. Manuscript, 2014.

10 Liming Cai and Jianer Chen. On the amount of nondeterminism and the power of verifying. *SIAM J. Comput.*, 26(3):733–750, 1997. doi:10.1137/S0097539793258295.

11 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *IWPEC 2009: Parameterized and Exact Computation*, volume 5917 of *LNCS*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0_6.

12 Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1335–1348. Association for Computing Machinery, 2020. doi:10.1145/3357713.3384283.

13 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC 1971, pages 151–158. Association for Computing Machinery, 1971. doi:10.1145/800157.805047.

14 Stephen A. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26(5):269–270, 1988. doi:10.1016/0020-0190(88)90152-4.

15 Stephen A. Cook and Robert A. Reckhow. Time bounded random access machines. *Journal of Computer and System Sciences*, 7(4):354–375, 1973. doi:10.1016/S0022-0000(73)80029-7.

16 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 J. Díaz and J. Torán. Classes of bounded nondeterminism. *Mathematical Systems Theory*, 23(1):21–32, 1990. doi:10.1007/BF02090764.

18 Graham E. Farr. *Topics in computational complexity.* PhD thesis, University of Oxford, 1986. URL: https://ora.ox.ac.uk/objects/uuid:ad3ed1a4-fea4-4b46-8e7a-a0c6a3451325/.

19 Uriel Feige and Joe Kilian. On Limited versus Polynomial Nondeterminism. *Chicago Journal of Theoretical Computer Science*, 1997(1), March 1997. URL: http://cjtcs.cs.uchicago.edu/articles/1997/1/cj97-01.pdf.

20 Lance Fortnow and Rahul Santhanam. New Non-Uniform Lower Bounds for Uniform Classes. In Ran Raz, editor, *CCC 2016: 31st Conference on Computational Complexity*, volume 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.19.

21 Judy Goldsmith, Matthew A. Levy, and Martin Mundhenk. Limited nondeterminism. *SIGACT News*, 27(2):20–29, 1996. doi:10.1145/235767.235769.

22 Yuri Gurevich and Saharon Shelah. Nearly linear time. In Albert R. Meyer and Michael A. Taitslin, editors, *Logic at Botik 1989*, volume 363 of *LNCS*, pages 108–118. Springer, 1989. doi:10.1007/3-540-51237-3_10.

23 J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965. doi:10.1090/S0002-9947-1965-0170805-7.

24 F.C. Hennie. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578, 1965. doi:10.1016/S0019-9958(65)90399-2.

25 Steven Homer and Alan L. Selman. *Computability and Complexity Theory*. Springer, 2nd edition, 2011.

26 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

27 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

28 Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time with applications. In *Proceedings of the Annual IEEE Conference on Computational Complexity*, CCC 2012, pages 1–9, 2012. doi:10.1109/CCC.2012.44.

29 Chandra M. R. Kintala and Patrick C. Fischer. Computations with a restricted number of nondeterministic steps (extended abstract). In *Proceedings of the ninth annual ACM symposium on Theory of computing*, STOC 1977, pages 178–185. Association for Computing Machinery, 1977. doi:10.1145/800105.803407.

30 Chandra M. R. Kintala and Patrick C. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM J. Comput.*, 9(1):46–53, 1980. doi:10.1137/0209003.

31 Kojiro Kobayashi. On proving time constructibility of functions. *Theoretical Computer Science*, 35:215–225, 1985. doi:10.1016/0304-3975(85)90015-5.

32 Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

33 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, pages 41–71, 2011.

34 Wolfgang Maass. Quadratic lower bounds for deterministic and nondeterministic one-tape turing machines. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC 1984, pages 401–408. Association for Computing Machinery, 1984. doi:10.1145/800057.808706.

35 Xu Mei-rui, John E. Doner, and Ronald V. Book. Refining nondeterminism in relativizations of complexity classes. *J. ACM*, 30(3):677—-685, 1983. doi:10.1145/2402.322399.

36 Ramamohan Paturi and Pavel Pudlak. On the complexity of circuit satisfiability. In *Proceedings of the Forty-second ACM symposium on Theory of computing*, STOC 2010, pages 241–250. ACM, 2010. doi:10.1145/1806689.1806724.

37 Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979. doi:10.1145/322123.322138.

38 Rahul Santhanam. On separators, segregators and time versus space. In *Proceedings of the Sixteenth Annual Conference on Computational Complexity*, CCC 2001, pages 286–294, 2001. doi:10.1109/CCC.2001.933895.

39 Joseph Swernofsky and Michael Wehar. On the complexity of intersecting regular, context-free, and tree languages. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015: Automata, Languages, and Programming - 42nd International Colloquium, Proceedings, Part II*, volume 9135 of *LNCS*, pages 414–426. Springer, 2015. doi:10.1007/978-3-662-47666-6_33.

40 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009. doi:10.1016/j.ipl.2008.10.014.

41 Michael Wehar. *On the Complexity of Intersection Non-Emptiness Problems.* PhD thesis, State University of New York at Buffalo, 2016. URL: http://michaelwehar.com/documents/mwehar_dissertation.pdf.

42 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

43 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. doi:10.1137/10080703X.

44 Ryan Williams. Non-uniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.

45 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *IPEC 2015: Proc. of the 10th International Symposium on Parameterized and Exact Computation*, volume 43 of *LIPICs*, pages 17–29, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

NP-Completeness of Perfect Matching Index of Cubic Graphs

Martin Škoviera
Department of Computer Science, Comenius University, Bratislava, Slovakia

Peter Varša
Department of Computer Science, Comenius University, Bratislava, Slovakia

Abstract

The perfect matching index of a cubic graph G, denoted by $\pi(G)$, is the smallest number of perfect matchings needed to cover all the edges of G; it is correctly defined for every bridgeless cubic graph. The value of $\pi(G)$ is always at least 3, and if G has no 3-edge-colouring, then $\pi(G) \geq 4$. On the other hand, a long-standing conjecture of Berge suggests that $\pi(G)$ never exceeds 5. It was proved by Esperet and Mazzuoccolo [J. Graph Theory 77 (2014), 144–157] that it is NP-complete to decide for a 2-connected cubic graph whether $\pi(G) \leq 4$. A disadvantage of the proof (noted by the authors) is that the constructed graphs have 2-cuts. We show that small cuts can be avoided and that the problem remains NP-complete even for nontrivial snarks – cyclically 4-edge-connected cubic graphs of girth at least 5 with no 3-edge-colouring. Our proof significantly differs from the one due to Esperet and Mazzuoccolo in that it combines nowhere-zero flow methods with elements of projective geometry, without referring to perfect matchings explicitly.

2012 ACM Subject Classification Mathematics of computing → Matchings and factors; Theory of computation → Problems, reductions and completeness; Mathematics of computing → Graph coloring

Keywords and phrases cubic graph, edge colouring, snark, perfect matching, covering, NP-completeness

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.56

Funding This research was supported by the grants APVV-19-0308 and VEGA 1/0743/21.

Acknowledgements The authors wish to thank Edita Máčajová for helpful discusions.

1 Introduction

It is well known [7] that every bridgeless cubic graph has a perfect matching that contains an arbitrarily preassigned edge. As a consequence, each such graph can be expressed as a union of a collection of its perfect matchings. The smallest number of perfect matchings needed for this purpose is its *perfect matching index*, denoted by $\pi(G)$. Although no constant bound on $\pi(G)$ is known, a fascinating conjecture of Berge (see [8]) suggests that five perfect matchings should do for every bridgeless cubic graph G.

Clearly, $\pi(G) = 3$ if and only if G is 3-edge-colourable, so if G has chromatic index 4, the value of $\pi(G)$ is at least 4. Understanding the cubic graphs that require more than four perfect matchings to cover their edges is fundamental for any approach that might lead to proving or disproving Berge's conjecture. However, nontrivial examples of cubic graphs with perfect matching index at least 5 appear to be very rare and are difficult to find. In the list comprising all 64 326 024 *nontrivial snarks* – cyclically 4-edge-connected cubic graphs of girth at least 5 with no 3-edge-colouring – on up to 36 vertices, generated by Brinkmann et al. [2], there are only two graphs that cannot be covered with four perfect matchings: the Petersen graph and the *windmill snark* W_{34} on 34 vertices displayed in Figure 1. The latter snark provides the starting point for several infinite families of snarks with $\pi \geq 5$, see [1, 2, 3, 5].

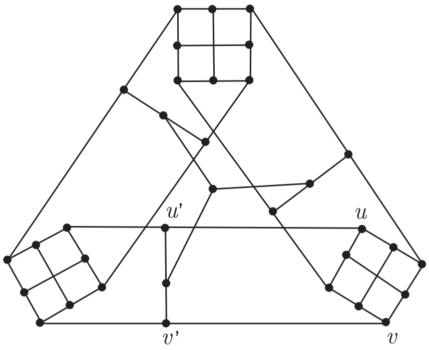


Figure 1 Windmill snark W_{34} on 34 vertices.

It transpires that the structure of graphs with perfect matching index at least 5 is far from being simple. In fact, deciding whether $\pi(G) \leq 4$ is an NP-complete problem, which was proved by Esperet and Mazzuoccolo [3] in 2014. However, as the authors write in [3], "*The gadgets used in the proof of NP-completeness have many* 2*-edge-cuts, so our* [first] *result does not say much about* 3*-edge-connected cubic graphs.*" In particular, they leave the NP-completeness problem open for nontrivial snarks. In this context it may be useful to realise that it is the class of nontrivial snarks which is particularly important for the problem. Indeed, several profound conjectures in graph theory, including the celebrated cycle double cover conjecture and the shortest cycle cover conjecture (also known as the 7/5-conjecture), can be reduced to nontrivial snarks with perfect matching index at least 5, see Steffen [9, Theorem 3.1].

The purpose of this contribution is to prove that deciding whether $\pi(G) \leq 4$ remains NP-complete even in the family of nontrivial snarks. Like the proof of NP-completeness due to Esperet and Mazzuoccolo [3], our proof employs reduction to 3-edge-colourability, which is known to be NP-complete by a result of Holyer [4]. On the other hand, its characteristic feature consists in avoiding direct use of perfect matchings, replacing them with nowhere-zero flows possessing an additional geometric structure within the 3-dimensional projective space $\mathbb{P}_3(\mathbb{F}_2)$ over the 2-element field. Although our methods heavily depend on the theory of tetrahedral flows developed in [5], we include all the necessary definitions and results from [5] to make the present paper self-contained.

We finish this section with a brief list of basic definitions used throughout the paper.

Our graphs will be mostly cubic, simple, although parallel edges and loops are not automatically excluded. A *circuit* is a connected 2-regular graph. A graph G is said to be *cyclically k-edge-connected* if the removal of fewer than k edges from G cannot create a graph with at least two components containing circuits. An edge cut S in G that separates two circuits from each other is *cycle-separating.* A (proper) *edge-colouring* of a graph G is a mapping from the edge set of G to a set of colours such that adjacent edges receive distinct colours. A *k-edge-colouring* is an edge colouring using k colours. A 2-connected cubic graph that does not admit a 3-edge-colouring is called a *snark.* A snark is *nontrivial* if it is cyclically 4-edge-connected and has no circuits of length smaller than 5.

For for more details and general context we refer the reader to [5].

2 Main results

Our point of departure is the result of Esperet and Mazzuoccolo [3, Theorem 2] which establishes NP-completeness of deciding whether $\pi(G) \leq 4$ in the class of bridgeless cubic graphs. We briefly summarise their proof.

Let G be an arbitrary bridgeless cubic graph. Inflate every vertex of G to a triangle, thereby producing a graph G'. Next, construct a new cubic graph G'' as follows. Take the *Tietze graph* T, which arises from the Petersen graph by inflating one of the vertices to a triangle, and remove an edge e lying on the triangle of T. For each edge x lying on a triangle of G' take a copy T_x of $T - e$, remove x from G', and connect the two 2-valent vertices of $G' - x$ to the two 2-valent vertices of T_x in such a way that 3-regularity is restored. The graph G'' is now obtained by repeating the just described procedure with each edge of G' lying on a triangle, see Figure 2. The substantial part of the proof of NP-completeness presented in [3] consists in checking that $\pi(G'') = 4$ if and only if G is 3-edge-colourable.

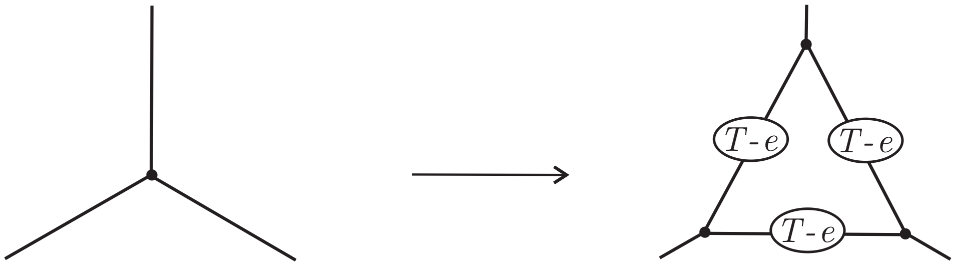


Figure 2 The construction of Esperet and Mazzuoccolo; $T - e$ denotes the Tietze graph with one edge removed.

.

Clearly, each copy T_x of $T - e$ is separated from the rest of G'' by a 2-edge-cut, so G'' has a plenty of 2-cuts. Considering the importance of nontrivial snarks for Berge's conjecture and other related conjectures it is a legitimate question to ask whether small cuts in the proof of NP-completeness can be avoided. Our main result answers this question in the positive.

▶ **Theorem 1.** *Deciding whether a nontrivial snark G satisfies $\pi(G) \leq 4$ is an* NP*-complete problem.*

The previous theorem is a direct consequence of the following more detailed statement.

▶ **Theorem 2.** *For every 2-connected cubic graph G of order n one can construct a derived graph $G^\sharp$ on $102n$ vertices which is a nontrivial snark. Moreover, $\pi(G^\sharp) = 4$ if and only if G is 3-edge-colourable.*

The derived graph $G^\sharp$ will be constructed by substituting the vertices of G with "fat vertices" (vertex gadgets, which we call *tripoles*) and the edges of G with "fat edges" (edge gadgets, which we call *dipoles*). Dipoles and tripoles, and more generally *multipoles*, are structures similar to graphs: like graphs, they consist of vertices and edges, each edge having two *half-edges*. In addition to proper edges, multipoles may contain *dangling edges*, with only one half-edge incident with a vertex, and even *isolated edges*, which are not incident with any vertex at all. Thus a dangling edge has one free half-edge while an isolated edge has two free half-edges. A *dipole* is a multipole whose free half-edges are partitioned into two subsets, the *input connector* and the *output connector*, while a *tripole* has its free half-edges distributed into three *connectors*. All multipoles in this paper are *cubic*, that is to say, every vertex is incident with exactly three half-edges. Moreover, all connectors will be of size 2.

We now describe the conctruction of $G^\sharp$ in detail. Let G be an arbitrary 2-connected cubic graph.

For vertex gadgets we use copies of the tripole W_0 consisting of three isolated edges a, b, and c (and no vertices) in which each connector consists of half-edges that belong to two distinct edges; the tripole W_0 is shown in Figure 3.

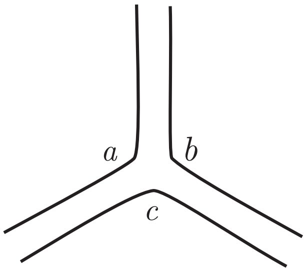


Figure 3 The tripole W_0.

For edge gadgets we take copies of a dipole H_{68} on 68 vertices; it is somewhat more complicated to describe. First, take the windmill snark W_{34} of order 34 shown in Figure 1 and severe the edges uu' and vv' indicated in the figure thereby producing four dangling edges distributed into two pairs. The resulting dipole, which we denote by D_{34}, is displayed in Figure 4. The input connector $\{d, k\}$ is formed from the half-edges belonging to the dangling edges incident with u and v, respectively, while the output connector $\{q, m\}$ is formed from those incident with u' and v'. To finish the construction of H_{68}, take two copies of D_{34} and weld the half-edges of their input connectors identically, that is to say, join d to d and k to k. The result is the dipole on 68 vertices whose both connectors are copies of the output connector of D_{34} – this is the required H_{68}.

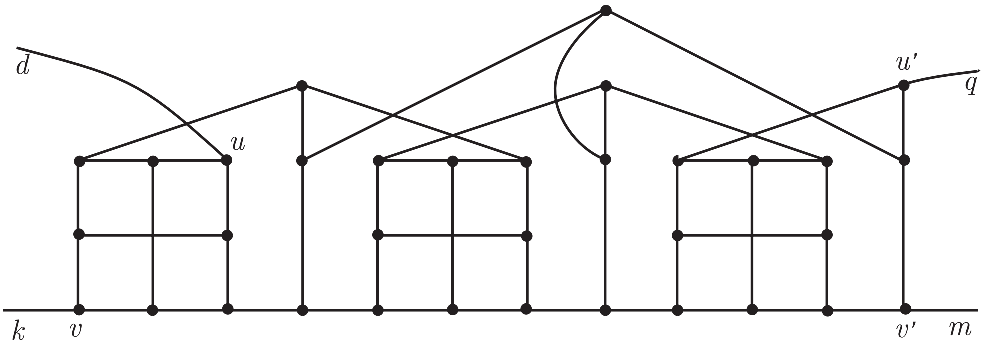


Figure 4 The dipole D_{34}.

Finally, we assemble $G^\sharp$ from the building blocks.

1. For each vertex v of G we take a cyclic permutation R_v of the edges incident with v. The collection $R = (R_v)_{v \in V(G)}$ is the *rotation system* for G, as it is generally known in topological graph theory [6]. The choice of R is irrelevant, yet R is important for keeping the track of the construction. (We will not pursue the topological connection any further. No knowledge of topological graph theory is therefore required.)
2. Next, we create the corresponding vertex gadget X_v as a copy of the tripole W_0. We associate each connector of X_v with an edge of G incident with v: for each such edge z we let $\{z', z''\}$ be the corresponding connector and we further require that the half-edges z' and $(R_v(z))''$ constitute one edge, see Figure 5.

3. For any edge $e = uv$ of G incident with v we create the corresponding edge gadget Y_e as a copy of H_{68} and associate its connectors with the endvertices of e.
4. Finally, we glue Y_e to X_v. If $\{q', m'\}$ is the connector of Y_e associated with v, we take the connector $\{e', e''\}$ of X_v associated with e and attach q' to e' and m' to e''. As soon as the gluing procedure is performed with each edge e and with both its endvertices, the construction of the derived graph $G^\sharp$ is completed. It is easy to see that if G has n vertices, then $G^\sharp$ has $102n$ vertices. It follows that $G^\sharp$ can be constructed from G in a polynomial number of steps with respect to the number of vertices of G.

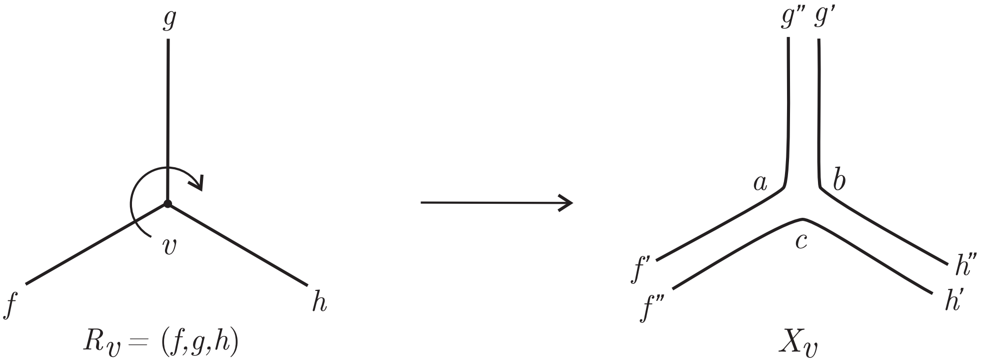


Figure 5 Substituting a vertex of G with a vertex gadget.

Recall that deciding whether a cubic graph is 3-edge-colourable is a well known NP-complete problem [4]. Therefore, from the construction of $G^\sharp$ it is clear that to prove Theorem 1 it suffices to show that $\pi(G^\sharp) = 4$ if and only if G is 3-edge-colourable.

3 Geometric background

In this section we prepare the machinery required for the proof of Theorem 1. As already indicated, the main idea of our proof consists in representing covers of a cubic graph with four perfect matchings by flows possessing a certain geometric structure. We now explain this idea in detail.

First of all, recall that a *flow* on a graph G is a function $\phi\colon E(G) \to A$, with values in an abelian group A, together with an orientation of G, such that the following property is fulfilled: at each vertex of G the sum of all incoming values equals the sum of all outgoing ones (*Kirchhoff's law*). More specifically, ϕ is an *A-flow.* A flow ϕ is *nowhere-zero* if $\phi(e) \neq 0$ for each edge e of G. The choice of an orientation for a flow is immaterial because the orientation of any edge can be reversed and its value can be replaced with the inverse without violating the Kirchhoff law. Furthermore, if $x = -x$ for every $x \in A$, one can ignore orientation altogether. This is possible precisely when A is isomorphic to an elementary abelian 2-group $\mathbb{Z}_2^n$.

Let G be a cubic graph that admits a covering $\mathcal{C} = \{P_1, P_2, P_3, P_4\}$ of its edges with four perfect matchings; note that the matchings need not be pairwise distinct. Clearly, $\mathcal{C}$ can be unambiguously represented by the mapping

$$\xi_{\mathcal{C}}\colon E(G) \to \mathbb{Z}_2^4$$

where the i-th coordinate of $\xi_{\mathcal{C}}(e)$ equals $1 \in \mathbb{Z}_2$ whenever the edge e does not belong to the perfect matching P_i. It is not difficult to see that $\xi_{\mathcal{C}}$ is a nowhere-zero $\mathbb{Z}_2^4$-flow on G.

In order to reveal important properties of this flow it is convenient to identify the set $\mathbb{Z}_2^4 - \{0\}$ with the point set of the 3-dimensional projective space $PG(3,2)$ over the 2-element field. Recall that the *n-dimensional projective space* $PG(n,2) = \mathbb{P}_n(\mathbb{F}_2)$ over the 2-element field $\mathbb{F}_2$ is an incidence geometry whose *points* can be identified with the nonzero vectors of the $(n+1)$-dimensional vector space $\mathbb{F}_2^{n+1}$ and whose *lines* are formed by the triples $\{x,y,z\}$ of points such that $x+y+z=0$. The 3-*dimensional projective space* $PG(3,2)$ consists of 15 points and 35 lines. The crucial observation concerning the flow ξ_C is that for every vertex v of G the values assigned by ξ_C to the edges incident with v form a line of $PG(3,2)$. The theory which we now outline generalises this observation.

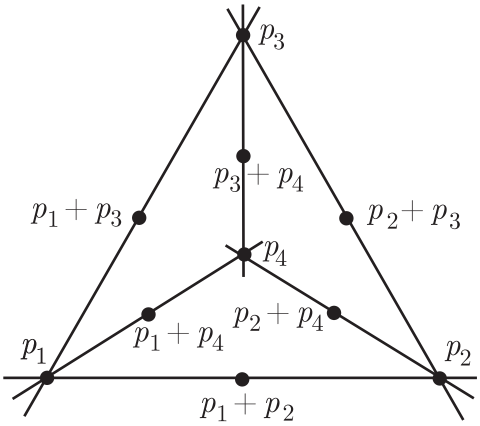


Figure 6 The tetrahedron in $PG(3,2)$ spanned by points p_1, p_2, p_3, and p_4.

We start with the necessary geometric concepts. A *tetrahedron* $T = T(p_1,p_2,p_3,p_4)$ in $PG(3,2)$ is a configuration consisting of ten points and six lines spanned by a set $\{p_1,p_2,p_3,p_4\}$ of four points of $PG(3,2)$ in general position; the latter means that the set $\{p_1,p_2,p_3,p_4\}$ constitutes a basis of the vector space $\mathbb{F}_2^4$. These four points are the *corner points* of T. Any two distinct corner points $c_1, c_2 \in \{p_1,p_2,p_3,p_4\}$ belong to a unique line $\ell = \{c_1, c_2, c_1+c_2\}$ of T whose third point c_1+c_2 is the *midpoint* of ℓ. Each line of T is uniquely determined by its midpoint. The tetrahedron $T(p_1,p_2,p_3,p_4)$ is depicted in Figure 6.

Given a tetrahedron T, a *T-flow* on a cubic graph G is a mapping $\phi\colon E(G) \to P(T)$ from the edge set of G to the *point set* $P(T)$ of T such that for each vertex v of G the three edges e_1, e_2, and e_3 incident with v receive values that form a line of T. The latter means that $\phi(e_1)+\phi(e_2)+\phi(e_3)=0$, which amounts to the Kirchhoff law for ϕ. Thus a T-flow is indeed a flow. A *tetrahedral flow* on G is a T-flow for some tetrahedron T in $PG(3,2)$. Note that any tetrahedal flow is also a proper edge colouring, which is why we occasionally refer to a T-flow as a colouring.

The next theorem provides a characterisation of cubic graphs with perfect matching index at most 4 in terms of tetrahedral flows. Due to the result of Esperet and Mazzuocollo [3], this characterisation is not efficient in the strict algorithmic sense, nevertheless, it is very useful.

▶ **Theorem 3.** *A cubic graph G can have its edges covered with four perfect matchings if and only if it admits a tetrahedral flow. Moreover, there exists a one-to-one correspondence between coverings of G with four perfect matchings and T-flows, where T is an arbitrary fixed tetrahedron in $PG(3,2)$.*

The way in which we apply Theorem 3 to the investigation of cubic graphs with perfect matching index at least 5 is based on the following idea: Suppose that a graph G in question has a cycle-separating 4-edge-cut S. Severing the edges of S produces two multipoles X_1 and X_2 with four dangling edges each, which we arrange into dipoles correspondingly. Choosing an input connector in each of them permits us to analyse how pairs of points of a tetrahedron in $PG(3,2)$ are transformed via a tetrahedral flow from the input connector to the output, and to check whether the tetrahedral flows through X_1 are always in conflict with the tetrahedral flows through X_2. This is why we need to examine which pairs of points can occur on the connectors of a dipole equipped with a tetrahedral flow.

For the rest of this section fix an arbitrary tetrahedron $T(p_1,p_2,p_3,p_4) = T$ in $PG(3,2)$. Consider a dipole $X = X(I,O)$ with input connector $I = \{g_1,g_2\}$ and output connector $O = \{h_1,h_2\}$. Given subsets $\{x,y\}$ and $\{x',y'\}$ of $P(T)$, we say that X *has a transition*

$$\{x,y\} \to \{x',y'\}$$

or that $\{x,y\} \to \{x',y'\}$ is a *transition through* X, if there exists a T-flow ϕ on X such that $\{\phi(g_1),\phi(g_2)\} = \{x,y\}$ and $\{\phi(h_1),\phi(h_2)\} = \{x',y'\}$. By the Kirchhoff law, for each transition $\{x,y\} \to \{x',y'\}$ through X we have $x+y = x'+y'$. This common value is called the *trace* of the transition; it can be any element of $\mathbb{Z}_2^4$.

In order to get better insight into possible transitions through a dipole it is useful to classify pairs of points of T according to their geometric shape. We say that two sets A and B of points of a tetrahedron T have the *same shape* if there exists a collineation (in other words, an automorphism) of $PG(3,2)$ that preserves T and takes A to B. A *geometric shape*, or simply a *shape*, is an equivalence class of all point sets having the same shape. The *shape* of a set of points of T is a geometric shape it belongs to. It is proved in [5] that each pair $\{x,y\}$ of points of T, where possibly $x = y$, falls into one of the following seven shapes: *line segment* `ls`, *half-line* `hl`, *angle* `ang`, *altitude* `alt`, *axis* `ax`, *double corner point* `dc`, and *double midpoint* `dm`. Their typical representatives are, respectively, the following pairs: $\{p_1,p_2\}$, $\{p_1,p_1+p_2\}$, $\{p_1+p_2,p_1+p_3\}$, $\{p_1,p_2+p_3\}$, $\{p_1+p_2,p_3+p_4\}$, $\{p_1,p_1\}$, and $\{p_1+p_2,p_1+p_2\}$. The set

$$\mathbf{\Sigma} = \{\mathtt{ls},\mathtt{hl},\mathtt{ang},\mathtt{alt},\mathtt{ax},\mathtt{dc},\mathtt{dm}\}$$

comprises all shapes of point pairs of T.

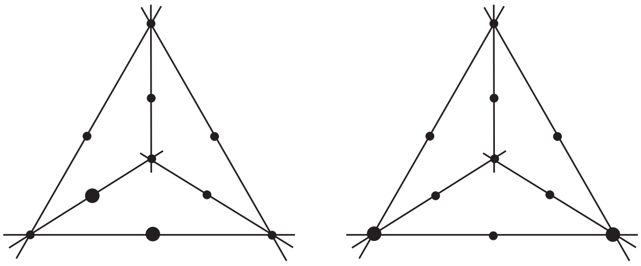


Figure 7 An angle (left) and a line segment (right).

Each transition $\{x,y\} \to \{x',y'\}$ through a dipole X between point pairs induces a transition between their shapes. To be more precise, for elements `s` and `t` of $\mathbf{\Sigma}$ we say that X *has a transition*

$$\mathtt{s} \to \mathtt{t}$$

if X has a transition $\{x,y\} \to \{x',y'\}$ such that `s` is the shape of $\{x,y\}$ and `t` is the shape of $\{x',y'\}$. It can be shown (see [5, Theorem 5.1]) that all transitions through any dipole have the form `s` $\to$ `s` except possibly the transitions `ls` $\to$ `ang` or `ang` $\to$ `ls`, and the transitions `dc` $\to$ `dm` and `dm` $\to$ `dc` between the degenerate point pairs. The two exceptional non-degenerate shapes `ang` and `ls` are illustrated in Figure 7.

For a detailed account of the theory of tetrahedral flows we refer the reader to [5].

4 The proof

In this section we prove Theorems 1 and 2. As previously mentioned, it suffices to prove the latter.

Our first step is to stablish several useful properties of the edge gadget, the dipole H_{68}. Recall that H_{68} consists of two copies of the dipole D_{34} whose input connectors have been identically glued together. The dipole D_{34} is the smallest example of what in [5] is called an *extended Halin dipole*. Every extended Halin dipole arises from a Halin snark (defined in [5]) by severing two edges in a manner similar to the construction of D_{34} from W_{34} (the latter being the smallest Halin snark). The crucial property of an extended Halin dipole, proved in [5, Theorem 8.8], is that every transition through it has the form

`ang` $\to$ `ls`.

One such transition through D_{34} is displayed in Figure 8. The flow is encoded as follows: the label i stands for the corner point p_i of the tetrahedron $T(p_1,p_2,p_3,p_4)$ while the label ij stands for the midpoint $p_i + p_j$. We will use this encoding in the rest of this paper.

The fact that `ang` $\to$ `ls` is the only possible transition of shapes through D_{34} can be checked directly, but without deeper involvement of the theory outlined in the previous section the proof would be quite tedious.

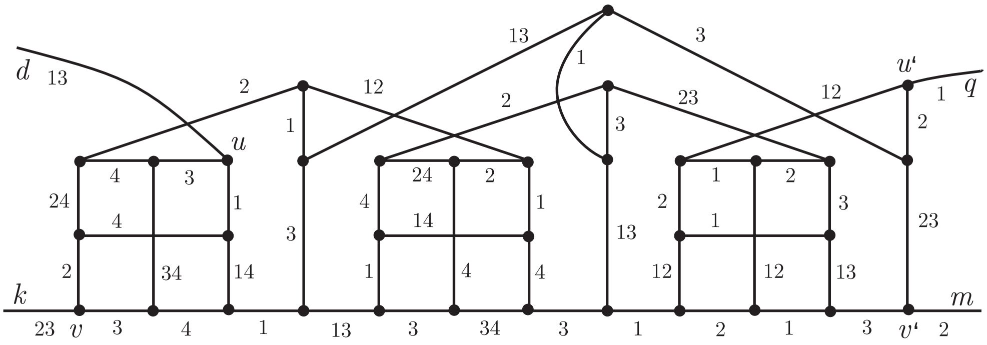


Figure 8 Transition `ang` $\to$ `ls` through D_{34}; encoding of colours: $i \mapsto p_i$, $ij \mapsto p_i + p_j$.

We need the following lemma.

▶ **Lemma 4.** *Every transition $\{x,y\} \to \{x',y'\}$ through the dipole H_{68} has the form*

`ls` $\to$ `ls`.

Such a transition exists for any line segment $\{x,y\}$ in T. Moreover, $\{x,y\} = \{x',y'\}$, and the two points may occur on the connectors of H_{68} in any order.

Proof. Recall that H_{68} is created from two copies of the dipole D_{34} whose input connectors are joined identically. Since $\mathtt{ang} \to \mathtt{ls}$ is the only possible transition through D_{34}, it follows that the only transition through H_{68} is of the form $\mathtt{ls} \to \mathtt{ls}$. If $\{x, y\} \to \{x', y'\}$ is any such transition, then $x + y = x' + y'$ by the Kirchhoff law. Each line of a tetrahedron is uniquely determined by its midpoint, so $x + y$ and $x' + y'$ must be midpoints of the same line, and in turn the line segments $\{x, y\}$ and $\{x', y'\}$ must be identical.

Now, let ψ^+ denote the flow displayed in Figure 8. In terms of ordered pairs of points, ψ^+ induces the transition $(p_1 + p_3, p_2 + p_3) \to (p_1, p_2)$, where the input pair stands for $(\psi^+(d), \psi^+(k))$ and the output pair stands for $(\psi^+(q), \psi^+(m))$. The dipole D_{34} has another tetrahedral flow ψ^-, namely one that represents the transition $(p_1 + p_3, p_2 + p_3) \to (p_2, p_1)$ with the output values swapped. This flow can easily be obtained from ψ^+ by interchanging p_1 and p_2 on the unique path in D_{34} which starts with the half-edge q of the output connector, leads through two internal edges, one incident with u' and the other incident with v', and terminates in the output connector with the half-edge m. The flows ψ^+ and ψ^- can be combined into four distinct tetrahedral flows on H_{68} which transform the line segment $\{p_1, p_2\}$ into itself in such a way that both the input pair and the output pair occur in any preassigned ordering. By symmetry, the same is true for any other line segment of T. The lemma follows. ◀

Now we are ready to prove Theorem 2.

Proof of Theorem 2. We first prove that $G^\sharp$ is not 3-edge-colourable irrespectively of the choice of G. Observe that in the language of tetrahedral flows a cubic graph is 3-edge-colourable if and only if it admits a tetrahedral flow using a single line of the tetrahedron. As we already know, every tetrahedral flow on D_{34} induces a transition of the form $\mathtt{ang} \to \mathtt{ls}$. Since the points of an angle do not lie on the same line of T, every tetrahedral flow on D_{34} must use points of at least two lines of the tetrahedron. Thus D_{34} is not 3-edge-colourable, and consequently neither is $G^\sharp$.

Next we prove that $G^\sharp$ is a nontrivial snark. Obviously, the girth of $G^\sharp$ is 5. To see that G is cyclically 4-edge-connected it is sufficient to realise that the underlying graph G is 2-connected and that each edge gadget arises from a cyclically 4-edge-connected cubic graphs by severing two independent edges. A straightforward case analysis, which we leave to the reader, shows that $G^\sharp$ has no k-edge-cut with $k < 4$ that separates a subgraph containing a cycle from the rest of $G^\sharp$. Summing up, $G^\sharp$ is a nontrivial snark.

We proceed to proving that $\pi(G^\sharp) = 4$ if and only if G is 3-edge-colourable. We do it in two steps.

▷ Claim. If $\pi(G^\sharp) = 4$, then G is 3-edge-colourable.

Proof. Assume that $\pi(G^\sharp) = 4$. By Theorem 3, $G^\sharp$ admits a T-flow ϕ where $T = T(p_1, p_2, p_3, p_4)$. For each edge e of G let $\phi'(e)$ denote the trace of the transition through the edge gadget Y_e of $G^\sharp$ induced by ϕ. Since all edge gadgets of $G^\sharp$ are copies of H_{68}, for each edge e of G the value $\phi'(e)$ is a midpoint of T.

Consider an arbitrary vertex v of G, and let e_1, e_2, and e_3 be the edges incident with v. By Kirchhoff's law, the outflow from the vertex gadget X_v must be 0, which in turn implies that $\phi'(e_1) + \phi'(e_2) + \phi'(e_3) = 0$. The values $\phi'(e_1)$, $\phi'(e_2)$, and $\phi'(e_3)$ are nonzero and therefore pairwise distinct. It follows that $\phi' \colon E(G) \to \mathbb{Z}_2^4$ is a nowhere-zero flow and the same time a proper edge colouring. As a colouring, ϕ uses (at most) six colours, the midpoints of T.

Next we prove that $\phi'(e_1)$, $\phi'(e_2)$, and $\phi'(e_3)$ are midpoints of the same triangle of T. By a *triangle* we mean a configuration of three lines of T spanned by three distinct corner points of T. Clearly, three distinct lines of T form a triangle if and only if any two of them intersect, but the intersection of all three is empty. Set $\phi'(e_i) = m_i$ for each $i \in \{1, 2, 3\}$; as already mentioned, m_1, m_2, and m_3 are pairwise distinct. Let ℓ_i denote the unique line of T containing m_i. We first prove that any two of the lines ℓ_1, ℓ_2, and ℓ_3 intersect. Suppose not, and assume that, say, ℓ_1 and ℓ_2 are disjoint. The position of any pair of disjoint lines of T implies that exists a permutation σ of $\{1, 2, 3, 4\}$ such that ℓ_1 is the line through $p_{\sigma(1)}$ and $p_{\sigma(2)}$, and ℓ_2 is the line through $p_{\sigma(3)}$ and $p_{\sigma(4)}$. Since $m_1 + m_2 + m_3 = 0$, we conclude that

$$m_3 = p_{\sigma(1)} + p_{\sigma(2)} + p_{\sigma(3)} + p_{\sigma(4)} = p_1 + p_2 + p_3 + p_4,$$

which is not a midpoint of any line of T. Therefore ℓ_1, ℓ_2, and ℓ_3 are distinct pairwise intersecting lines. Next we prove that $\ell_1 \cap \ell_2 \cap \ell_3 = \emptyset$. Indeed, if there is a point $p \in \ell_1 \cap \ell_2 \cap \ell_3$, then p is a corner point of T and ℓ_1, ℓ_2, and ℓ_3 are the three lines of T containing p. But then $m_1 + m_2 + m_3 = p_1 + p_2 + p_3 + p_4 \neq 0$, which is impossible. The only remaining possibility is that ℓ_1, ℓ_2, and ℓ_3 form a triangle. The latter means that there exist three distinct corner points c_1, c_2, and c_3 of T such that $m_1 = c_2 + c_3$, $m_2 = c_1 + c_3$, and $m_3 = c_1 + c_2$.

Now we are ready to produce a proper 3-edge-colouring of G. Let us define the mapping ψ from the set of all midpoints of T to the set $\{1, 2, 3\}$ as follows:

$$\begin{aligned} p_1 + p_2,\ p_3 + p_4 &\quad\mapsto\quad 1,\\ p_1 + p_3,\ p_2 + p_4 &\quad\mapsto\quad 2,\\ p_1 + p_4,\ p_2 + p_3 &\quad\mapsto\quad 3. \end{aligned}$$

Since every triangle is determined by three distinct corner points c_1, c_2 and c_3 of T, its midpoints $c_1 + c_2$, $c_2 + c_3$, and $c_1 + c_3$ receive from ψ three distinct values. In other words, $\psi\phi'$ is a proper 3-edge-colouring of G, which establishes the claim. ◁

▷ Claim. If G is 3-edge-colourable, then $\pi(G^\sharp) = 4$.

Proof. Assume that G is 3-edge-colourable. By Theorem 3, it is sufficient to find a tetrahedral flow on $G^\sharp$. Our aim is to construct a tetrahedral flow $\gamma^\sharp$ of $G^\sharp$ by departing from a 3-edge-colouring $\gamma\colon E(G) \to \{1, 2, 3\}$.

First of all, we colour the vertex gadgets. With respect to the chosen rotation system R for G the vertices of G fall into two types depending on whether the cyclic order of colours around the vertex is $(1, 2, 3)$ (*Type 1*) or $(1, 3, 2)$ (*Type 2*). Consider a vertex v of G, which is incident with edges f, g, and h, and let X_v be the corresponding vertex gadget. Recall that the connectors of X_v are $\{f', f''\}$, $\{g', g''\}$, and $\{h', h''\}$, and that for each edge x of G incident with v, the free half-edges x' and $(R(x))''$ constitute one edge of X_v, see Figure 5. Let a, b, and c be the edges of X_v that have the half-edges f', g', and h', respectively.

Without loss of generality we may assume that $\gamma(f) = 1$. We intend to colour the edges of X_v with three distinct corner points of the tetrahedron T, say p_1, p_2, and p_3, in such a way that the connector $\{f', f''\}$ receives colours from the line segment $\{p_2, p_3\}$. This choice implies that under the colouring $\gamma^\sharp$ the edge b must receive colour p_1. The colours of the remaining two edges will depend on the type of v. If v is Type 1, then $\gamma(g) = 2$, and $\gamma(h) = 3$, and we set $\gamma^\sharp(a) = p_3$, $\gamma^\sharp(b) = p_1$, and $\gamma^\sharp(c) = p_2$, see Figure 9. If v is Type 2, then $\gamma(g) = 3$, and $\gamma(h) = 2$, and we set $\gamma^\sharp(a) = p_2$, $\gamma^\sharp(b) = p_1$, and $\gamma^\sharp(c) = p_3$, see Figure 10. We have thus coloured every vertex gadget of X_v in such a way that the connector $\{x', x''\}$ of X_v corresponding to the edge x of G incident with v receives colours from the line segment $\{p_1, p_2, p_3\} - \{p_i\}$ if and only if $\gamma(x) = i \in \{1, 2, 3\}$.

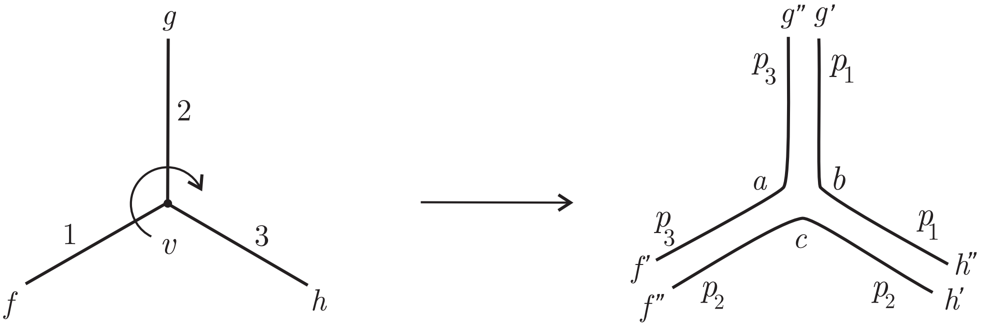


Figure 9 Colouring a vertex gadget that corresponds to a vertex of Type 1.

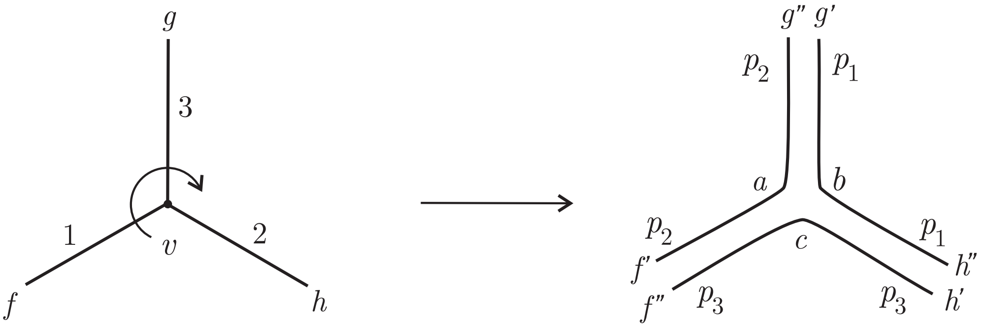


Figure 10 Colouring a vertex gadget that corresponds to a vertex of Type 2.

Next we colour the edge gadgets of $G^\sharp$. Consider an arbitrary edge $x = uv$ of G of colour, say, $\gamma(x) = 1$. We know that the half-edges in the connectors of both X_u and X_v corresponding to x receive colours from the line segment $\{p_2, p_3\}$. We now choose the tetrahedral colouring for the edge gadget Y_x which transforms the line segment $\{p_2, p_3\}$ into itself in such a way that the ordering of colours in both the input and the output of Y_x fits the ordering in the corresponding connectors of X_u and X_v. As argued in Lemma 4, such a colouring always exists. For edges of colours 2 and 3 we proceed analogously. In this way we transform a 3-edge-colouring γ of G into a tetrahedral colouring (that is, a tetrahedral flow) of $G^\sharp$. By Theorem 3, $\pi(G^\sharp) = 4$. This completes the proof of the claim as well as that of Theorem 2. ◁

◀

5 Final remark

The statement of Theorem 2 implies that the derived graph $G^\sharp$ is cyclically 4-edge-connected. If we relax cyclic 4-connectivity to 2-connectivity or 3-connectivity, the corresponding statement becomes significantly easier to prove. Indeed, take an arbitrary 2-edge-connected cubic graph G and substitute each vertex v with a copy X_v of the 3-pole Q obtained from the Petersen graph by removing a vertex. Identify the dangling edges of X_v with the edges of G incident with v, thereby producing a 2-connected cubic graph G^+; if G is 3-connected, so is G^+. Since Q is uncolourable, G^+ is a snark, though a trivial one. By employing our geometric theory it can be proved that $\pi(G^+) = 4$ if and only if G is 3-edge-colourable.

References

1 M. Abreu, T. Kaiser, D. Labbate, and G. Mazzuoccolo. Treelike snarks. *Electron. J. Combin.*, 23:#P3.54, 2016.

2 G. Brinkmann, J. Goedgebeur, J. Hägglund, and K. Markström. Generation and properties of snarks. *J. Combin. Theory Ser. B*, 103:468–488, 2013.

3 L. Esperet and G. Mazzuoccolo. On cubic bridgeless graphs whose edge-set cannot be covered by four perfect matchings. *J. Graph Theory*, 77:144–157, 2014.

4 I. Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10:718–720, 1981.

5 E. Máčajová and M. Škoviera. Cubic graphs that cannot be covered with four perfect matchings. *J. Combin. Theory Ser. B*, 150:144–176, 2021.

6 B. Mohar and C. Thomassen. *Graphs on Surfaces.* Johns Hopkins University Press, 2001.

7 J. Plesník. Connectivity of regular graphs and the existence of 1-factors. *Mat. Čas.*, 22:310–318, 1972.

8 P. D. Seymour. On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. *Proc. London Math. Soc.*, 38:423–460, 1979.

9 E. Steffen. 1-Factor and cycle covers of cubic graphs. *J. Graph Theory*, 78:195–206, 2015.

Optimal Oracles for Point-To-Set Principles

D. M. Stull
Northwestern University, Evanston, IL, USA

Abstract

The point-to-set principle [14] characterizes the Hausdorff dimension of a subset $E \subseteq \mathbb{R}^n$ by the *effective* (or algorithmic) dimension of its individual points. This characterization has been used to prove several results in classical, i.e., without any computability requirements, analysis. Recent work has shown that algorithmic techniques can be fruitfully applied to Marstrand's projection theorem, a fundamental result in fractal geometry.

In this paper, we introduce an extension of point-to-set principle - the notion of *optimal oracles* for subsets $E \subseteq \mathbb{R}^n$. One of the primary motivations of this definition is that, if E has optimal oracles, then the conclusion of Marstrand's projection theorem holds for E. We show that every analytic set has optimal oracles. We also prove that if the Hausdorff and packing dimensions of E agree, then E has optimal oracles. Moreover, we show that the existence of sufficiently nice outer measures on E implies the existence of optimal Hausdorff oracles. In particular, the existence of exact gauge functions for a set E is sufficient for the existence of optimal Hausdorff oracles, and is therefore sufficient for Marstrand's theorem. Thus, the existence of optimal oracles extends the currently known sufficient conditions for Marstrand's theorem to hold.

Under certain assumptions, every set has optimal oracles. However, assuming the axiom of choice and the continuum hypothesis, we construct sets which do not have optimal oracles. This construction naturally leads to a generalization of Davies' theorem on projections.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Algorithmic randomness, Kolmogorov complexity, geometric measure theory

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.57

Acknowledgements I would like to thank Denis Hirschfeldt, Jack Lutz and Chris Porter for very valuable discussions and suggestions. I would also like to thank the participants of the recent AIM workshop on Algorithmic Randomness.

1 Introduction

Effective, i.e., algorithmic, dimensions were introduced [12, 1] to study the randomness of points in Euclidean space. The effective dimension, $\dim(x)$ and effective strong dimension, $\mathrm{Dim}(x)$, are real values which measure the asymptotic density of information of an *individual point* x. The connection between effective dimensions and the classical Hausdorff and packing dimension is given by the point-to-set principle of J. Lutz and N. Lutz [14]: For any $E \subseteq \mathbb{R}^n$,

$$\dim_H(E) = \min_{A \subseteq \mathbb{N}} \sup_{x \in E} \dim^A(x), \text{ and} \tag{1}$$

$$\dim_P(E) = \min_{A \subseteq \mathbb{N}} \sup_{x \in E} \mathrm{Dim}^A(x)\,. \tag{2}$$

Call an oracle A satisfying (1) a *Hausdorff oracle* for E. Similarly, we call an oracle A satisfying (2) a *packing oracle* for E. Thus, the point-to-set principle shows that the classical notion of Hausdorff or packing dimension is completely characterized by the effective dimension of its individual points, relative to a Hausdorff or packing oracle, respectively.

Recent work as shown that algorithmic dimensions are not only useful in effective settings, but, via the point-to-set principle, can be used to solve problems in geometric measure theory [15, 17, 18, 19, 30]. It is important to note that the point-to-set principle allows one to use

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 57; pp. 57:1–57:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

algorithmic techniques to prove theorems whose statements have seemingly nothing to do with computability theory. In this paper, we focus on the connection between algorithmic dimension and Marstrand's projection theorem.

Marstrand, in his landmark paper [21], was the first to study how the dimension of a set is changed when projected onto a line. He showed that, for any *analytic* set $E \in \mathbb{R}^2$, for almost every angle $\theta \in [0, \pi)$,

$$\dim_H(p_\theta\, E) = \min\{\dim_H(E), 1\}, \tag{3}$$

where $p_\theta(x, y) = x \cos\theta + y \sin\theta$[1]. The study of projections has since become a central theme in fractal geometry (see [8] or [25] for a more detailed survey of this development).

Marstrand's theorem begs the question of whether the analytic requirement on E can be dropped. It is known that, without further conditions, it cannot. Davies [5] showed that, assuming the axiom of choice and the continuum hypothesis, there are non-analytic sets for which Marstrands conclusion fails. However, the problem of classifying the sets for which Marstrands theorem does hold is still open. Recently, Lutz and Stull [20] used the point-to-set principle to prove that the projection theorem holds for sets for which the Hausdorff and packing dimensions agree[2]. This expanded the reach of Marstrand's theorem, as this assumption is incomparable with analyticity.

In this paper, we give the broadest known sufficient condition (which makes essential use of computability theory) for Marstrand's theorem. In particular, we introduce the notion of *optimal Hausdorff oracles* for a set $E \subseteq \mathbb{R}^n$. We prove that Marstrand's theorem holds for every set E which has optimal Hausdorff oracles.

An optimal Hausdorff oracle for a set E is a Hausdorff oracle which minimizes the algorithmic complexity of "most"[3] points in E. It is not immediately clear that any set E has optimal oracles. Nevertheless, we show that two natural classes of sets $E \subseteq \mathbb{R}^n$ do have optimal oracles.

We show that every analytic, and therefore Borel, set has optimal oracles. We also prove that every set whose Hausdorff and packing dimensions agree has optimal Hausdorff oracles. Thus, we show that the existence of optimal oracles encapsulates the known conditions sufficient for Marstrand's theorem to hold. Moreover, we show that the existence of sufficiently nice outer measures on E implies the existence of optimal Hausdorff oracles. In particular, the existence of exact gauge functions (Section 2.1) for a set E is sufficient for the existence of optimal Hausdorff oracles for E, and is therefore sufficient for Marstrand's theorem. Thus, the existence of optimal Hausdorff oracles is weaker than the previously known conditions for Marstrand's theorem to hold.

We also show that the notion of optimal oracles gives insight to sets for which Marstrand's theorem does *not* hold. Assuming the axiom of choice and the continuum hypothesis, we construct sets which do not have optimal oracles. This construction, with minor adjustments, proves a generalization of Davies' theorem proving the existence of sets for which (3) does not hold. In addition, the inherently algorithmic aspect of the construction might be useful for proving set-theoretic properties of exceptional sets for Marstrand's theorem.

[1] This result was later generalized to $\mathbb{R}^n$, for arbitrary n, as well as extended to hyperspaces of dimension m, for any $1 \leq m \leq n$ (see e.g. [22, 23, 24]).

[2] Orponen [29] has recently given another proof of Lutz and Stull's result using more classical tools.

[3] By most, we mean a subset of E of the same Hausdorff dimension as E

Finally, we define optimal *packing* oracles for a set. We show that every analytic set E has optimal packing oracles. We also show that every E whose Hausdorff and packing dimensions agree have optimal packing oracles. Assuming the axiom of choice and the continuum hypothesis, we show that there are sets with optimal packing oracles without optimal Hausdorff oracles (and vice-versa).

The structure of the paper is as follows. In Section 2.1 we review the concepts of measure theory needed, and the (classical) definition of Hausdorff dimension. In Section 2.2 we review algorithmic information theory, including the formal definitions of effective dimensions. We then introduce and study the notion of optimal oracles in Section 3. In particular, we give a general condition for the existence of optimal oracles in Section 3.1. We use this condition to prove that analytic sets have optimal oracles in Section 3.2. We conclude in Section 3.3 with an example, assuming the axiom of choice and the continuum hypothesis, of a set without optimal oracles. The connection between Marstrands projection theorem and optimal oracles is explored in Section 4. In this section, we prove that Marstrands theorem holds for every set with optimal oracles. In Section 4.1, we use the construction of a set without optimal oracles to give a new, algorithmic, proof of Davies' theorem. Finally, in Sectino 5, we define and investigate the notion of optimal packing oracles.

2 Preliminaries

2.1 Outer Measures and Classical Dimension

A set function $\mu : \mathcal{P}(\mathbb{R}^n) \to [0, \infty]$ is called an *outer measure* on $\mathbb{R}^n$ if

1. $\mu(\emptyset) = 0$,
2. if $A \subseteq B$ then $\mu(A) \leq \mu(B)$, and
3. for any sequence $A_1, A_2, \ldots$ of subsets,

$$\mu(\bigcup_i A_i) \leq \sum_i \mu(A_i).$$

If μ is an outer measure, we say that a subset A is *μ-measurable* if

$$\mu(A \cap B) + \mu(B - A) = \mu(B),$$

for every subset $B \subseteq \mathbb{R}^n$.

An outer measure μ is called a *metric outer measure* if every Borel subset is μ-measurable and

$$\mu(A \cup B) = \mu(A) + \mu(B),$$

for every pair of subsets A, B which have positive Hausdorff distance. That is,

$$\inf\{\|x - y\| \mid x \in A, y \in B\} > 0.$$

An important example of a metric outer measure is the s-dimensional Hausdorff measure. For every $E \subseteq [0, 1)^n$, define the s-dimensional Hausdorff content at precision r by

$$h^s_r(E) = \inf\left\{\sum_i d(Q_i)^s \mid \bigcup_i Q_i \text{ covers } E \text{ and } d(Q_i) \leq 2^{-r}\right\},$$

where $d(Q)$ is the diameter of ball Q. We define the s-dimensional Hausdorff measure of E by

$$\mathcal{H}^s(E) = \lim_{r\to\infty} h^s_r(E).$$

▶ Remark 1. It is well-known that $\mathcal{H}^s$ is a metric outer measure for every s.

The *Hausdorff dimension* of a set E is then defined by

$$\dim_H(E) = \inf_s\{\mathcal{H}^s(E) = \infty\} = \sup_s\{\mathcal{H}^s(E) = 0\}.$$

Another important metric outer measure, which gives rise to the packing dimension of a set, is the s-dimensional packing measure. For every $E \subseteq [0,1)^n$, define the s-dimensional packing pre-measure by

$$p^s(E) = \limsup_{\delta\to 0}\left\{\sum_{i\in\mathbb{N}} d(B_i)^s \mid \{B_i\} \text{ is a set of disjoint balls and } B_i \in C(E,\delta)\right\},$$

where $C(E,\delta)$ is the set of all closed balls with diameter at most δ with centers in E. We define the s-dimensional packing measure of E by

$$\mathcal{P}^s(E) = inf\left\{\sum_j p^s(E_j) \mid E \subseteq \bigcup E_j\right\},$$

where the infimum is taken over all countable covers of E. For every s, the s-dimensional packing measure is a metric outer measure.

The *packing dimension* of a set E is then defined by

$$\dim_P(E) = \inf_s\{\mathcal{P}^s(E) = 0\} = \sup_s\{\mathcal{P}^s(E) = \infty\}.$$

In order to prove that every analytic set has optimal oracles, we will make use of the following facts of geometric measure theory (see, e.g., [7], [2]).

▶ **Theorem 1.** *The following are true.*

1. *Suppose $E \subseteq \mathbb{R}^n$ is compact and satisfies $\mathcal{H}^s(E) > 0$. Then there is a compact subset $F \subseteq E$ such that $0 < \mathcal{H}^s(F) < \infty$.*
2. *Every analytic set $E \subseteq \mathbb{R}^n$ has a Σ^0_2 subset $F \subseteq E$ such that $\dim_H(F) = \dim_H(E)$.*
3. *Suppose $E \subseteq \mathbb{R}^n$ is compact and satisfies $\mathcal{P}^s(E) > 0$. Then there is a compact subset $F \subseteq E$ such that $0 < \mathcal{P}^s(F) < \infty$.*
4. *Every analytic set $E \subseteq \mathbb{R}^n$ has a Σ^0_2 subset $F \subseteq E$ such that $\dim_P(F) = \dim_P(E)$.*

It is possible to generalize the definition of Hausdorff measure using gauge functions. A function $\phi : [0,\infty) \to [0,\infty)$ is a *gauge function* if ϕ is monotonically increasing, strictly increasing for $t > 0$ and continuous. If ϕ is a gauge, define the ϕ-Hausdorff content at precision r by

$$h^\phi_r(E) = \inf\left\{\sum_i \phi(d(Q_i)) \mid \bigcup_i Q_i \text{ covers } E \text{ and } d(Q_i) \leq 2^{-r}\right\},$$

where $d(Q)$ is the diameter of ball Q. We define the ϕ-Hausdorff measure of E by

$$\mathcal{H}^\phi(E) = \lim_{r\to\infty} h^\phi_r(E).$$

Thus we recover the s-dimensional Hausdorff measure when $\phi(t) = t^s$.

Gauged Hausdorff measures give fine-grained information about the size of a set. There are sets E which Hausdorff dimension s, but $\mathcal{H}^s(E) = 0$ or $\mathcal{H}^s(E) = \infty$. However, it is sometimes possible to find an appropriate gauge so that $0 < \mathcal{H}^\phi(E) < \infty$. When $0 < \mathcal{H}^\phi(E) < \infty$, we say that ϕ is an *exact gauge for E*.

▶ Example. For almost every Brownian path X in $\mathbb{R}^2$, $\mathcal{H}^2(X) = 0$, but $0 < \mathcal{H}^\phi(X) < \infty$, where $\phi(t) = t^2 \log \frac{1}{t} \log\log \frac{1}{t}$.

For two outer measures μ and ν, μ is said to be *absolutely continuous with respect to* ν, denoted $\mu \ll \nu$, if $\mu(A) = 0$ for every set A for which $\nu(A) = 0$.

▶ Example. For every s, let $\phi_s(t) = t^s \log \frac{1}{t}$. Then $\mathcal{H}^s \ll \mathcal{H}^{\phi_s}$.

▶ Example. For every s, let $\phi_s(t) = \frac{t^s}{\log \frac{1}{t}}$. Then $\mathcal{H}^{\phi_s} \ll \mathcal{H}^s$.

2.2 Algorithmic Information Theory

The *conditional Kolmogorov complexity* of a binary string $\sigma \in \{0,1\}^*$ given binary string $\tau \in \{0,1\}^*$ is

$$K(\sigma|\tau) = \min_{\pi \in \{0,1\}^*} \{\ell(\pi) : U(\pi, \tau) = \sigma\} ,$$

where U is a fixed universal prefix-free Turing machine and $\ell(\pi)$ is the length of π. The *Kolmogorov complexity* of σ is $K(\sigma) = K(\sigma|\lambda)$, where λ is the empty string. An important fact is that the choice of universal machine affects the Kolmogorov complexity by at most an additive constant (which, especially for our purposes, can be safely ignored). See [11, 28, 6] for a more comprehensive overview of Kolmogorov complexity.

We can naturally extend these definitions to Euclidean spaces by introducing "precision" parameters [16, 14]. Let $x \in \mathbb{R}^m$, and $r, s \in \mathbb{N}$. The *Kolmogorov complexity of x at precision r* is

$$K_r(x) = \min \{K(p) \ : \ p \in B_{2^{-r}}(x) \cap \mathbb{Q}^m\} .$$

The *conditional Kolmogorov complexity of x at precision r given* $q \in \mathbb{Q}^m$ is

$$\hat{K}_r(x|q) = \min \{K(p\,|\,q) \ : \ p \in B_{2^{-r}}(x) \cap \mathbb{Q}^m\} .$$

The *conditional Kolmogorov complexity of x at precision r given $y \in \mathbb{R}^n$ at precision s* is

$$K_{r,s}(x|y) = \max \left\{\hat{K}_r(x|q) \ : \ q \in B_{2^{-s}}(y) \cap \mathbb{Q}^n\right\} .$$

We typically abbreviate $K_{r,r}(x|y)$ by $K_r(x|y)$.

The *effective Hausdorff dimension* and *effective packing dimension*[4] of a point $x \in \mathbb{R}^n$ are

$$\dim(x) = \liminf_{r\to\infty} \frac{K_r(x)}{r} \quad \text{and} \quad \mathrm{Dim}(x) = \limsup_{r\to\infty} \frac{K_r(x)}{r} .$$

By letting the underlying fixed prefix-free Turing machine U be a universal *oracle* machine, we may *relativize* the definition in this section to an arbitrary oracle set $A \subseteq \mathbb{N}$. The definitions of $K^A_r(x)$, $\dim^A(x)$, $\mathrm{Dim}^A(x)$, etc. are then all identical to their unrelativized versions, except that U is given oracle access to A. Note that taking oracles as subsets of the naturals is quite general. We can, and frequently do, encode a point y into an oracle, and consider the complexity of a point *relative* to y. In these cases, we typically forgo explicitly referring to this encoding, and write e.g. $K^y_r(x)$. We can also *join* two oracles $A, B \subseteq \mathbb{N}$ using any computable bijection $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We denote the join of A and B by (A, B). We can generalize this procedure to join any countable sequence of oracles.

[4] Although effective Hausdorff was originally defined by J. Lutz [13] using martingales, it was later shown by Mayordomo [26] that the definition used here is equivalent. For more details on the history of connections between Hausdorff dimension and Kolmogorov complexity, see [6, 27].

As mentioned in the introduction, the connection between effective dimensions and the classical Hausdorff and packing dimensions is given by the point-to-set principle introduced by J. Lutz and N. Lutz [14].

▶ **Theorem 2** (Point-to-set principle). *Let $n \in \mathbb{N}$ and $E \subseteq \mathbb{R}^n$. Then*

$$\dim_H(E) = \min_{A \subseteq \mathbb{N}} \sup_{x \in E} \dim^A(x), \text{ and}$$

$$\dim_P(E) = \min_{A \subseteq \mathbb{N}} \sup_{x \in E} \operatorname{Dim}^A(x).$$

An oracle testifying to the the first equality is called a *Hausdorff oracle for E*. Similarly, an oracle testifying to the the second equality is called a *packing oracle for E*.

3 Optimal Hausdorff Oracles

For any set E, there are infinitely many Hausdorff oracles for E. A natural question is whether there is a Hausdorff oracle which minimizes the complexity of every point in E. Unfortunately, it is, in general, not possible for a single oracle to maximally reduce *every* point. We introduce the notion of optimal Hausdorff oracles by weakening the condition to a *single* point.

▶ **Definition 3.** *Let $E \subseteq \mathbb{R}^n$ and $A \subseteq \mathbb{N}$. We say that A is Hausdorff optimal for E if the following conditions are satisfied.*

1. *A is a Hausdorff oracle for E.*
2. *For every $B \subseteq \mathbb{N}$ and every $\epsilon > 0$ there is a point $x \in E$ such that $\dim^{A,B}(x) \geq \dim_H(E) - \epsilon$ and for almost every $r \in \mathbb{N}$*

$$K_r^{A,B}(x) \geq K_r^A(x) - \epsilon r.$$

Note that the second condition only guarantees the existence of *one* point whose complexity is unaffected by the addtional information in B. However, we can show that this implies the seemingly stronger condition that "most" points are unaffected. For $B \subseteq \mathbb{N}$, $\epsilon > 0$ define the set

$$N(A, B, \epsilon) = \{x \in E \mid (\forall^\infty r)\, K_r^{A,B}(x) \geq K_r^A(x) - \epsilon r\}.$$

▶ **Proposition 4.** *Let $E \subseteq \mathbb{R}^n$ be a set such that $\dim_H(E) > 0$ and let A be an oracle. Then A is a Hausdorff optimal oracle for E if and only if A is a Hausdorff oracle and $\dim_H(N(A, B, \epsilon)) = \dim_H(E)$ for every $B \subseteq \mathbb{N}$ and $\epsilon > 0$.*

A simple, but useful, result is if B is an oracle obtained by adding additional information to an optimal Hausdorff oracle, then B is also optimal.

▶ **Lemma 5.** *Let $E \subseteq \mathbb{R}^n$. If A is an optimal Hausdorff oracle for E, then the join $C = (A, B)$ is Hausdorff optimal for E for every oracle B.*

We now give some basic closure properties of the class of sets with optimal Hausdorff oracles.

▶ **Observation 6.** *Let $F \subseteq E$. If $\dim_H(F) = \dim_H(E)$ and F has an optimal Hausdorff oracle, then E has an optimal Hausdorff oracle.*

We can also show that having optimal Hausdorff oracles is closed under countable unions.

▶ **Proposition 7.** *Let $E_1, E_2, \ldots$ be a countable sequence of sets and let $E = \cup_n E_n$. If every set E_n has an optimal Hausdorff oracle, then E has an optimal Hausdorff oracle.*

3.1 Outer Measures and Optimal Oracles

In this section we give a sufficient condition for a set to have optimal Hausdorff oracles. Specifically, we prove that if $\dim_H(E) = s$, and there is a metric outer measure, absolutely continuous with respect to $\mathcal{H}^s$, such that $0 < \mu(E) < \infty$, then E has optimal Hausdorff oracles. Although stated in this general form, the main application of this result (in Section 3.2) is for the case $\mu = \mathcal{H}^s$.

For every $r \in \mathbb{N}$, let $\mathcal{Q}^n_r$ be the set of all dyadic cubes at precision r, i.e., cubes of the form

$$Q = [m_1 2^{-r}, (m_1 + 1)2^{-r}) \times \ldots \times [m_n 2^{-r}, (m_n + 1)2^{-r}),$$

where $0 \leq m_1, \ldots, m_n \leq 2^r$. For each r, we refer to the 2^{nr} cubes in $\mathcal{Q}_r$ as $Q_{r,1}, \ldots, Q_{r,2^{nr}}$. We can identify each dyadic cube $Q_{r,i}$ with the unique dyadic rational $d_{r,i}$ at the center of $Q_{r,i}$.

We now associate, to each metric outer measure, a *discrete semimeasure on the dyadic rationals* $\mathbb{D}$. Recall that discrete semimeasure on $\mathbb{D}^n$ is a function $p : \mathbb{D}^n \to [0, 1]$ which satisfies $\Sigma_{r,i} p(d_{r,i}) < \infty$.

Let $E \subseteq \mathbb{R}^n$ and μ be a metric outer measure such that $0 < \mu(E) < \infty$. Define the function $p_\mu : \mathbb{D}^n \to [0, 1]$ by

$$p_{\mu,E}(d_{r,i}) = \frac{\mu(E \cap Q_{r,i})}{r^2 \mu(E)}.$$

▶ **Observation 8.** *Let μ be a metric outer measure and $E \subseteq \mathbb{R}^n$ such that $0 < \mu(E) < \infty$. Then for every r, every dyadic cube $Q \in \mathcal{Q}_r$, and all $r' > r$,*

$$\mu(E \cap Q) = \sum_{\substack{Q' \subset Q \\ Q' \in \mathcal{Q}_{r'}}} \mu(E \cap Q').$$

▶ **Proposition 9.** *Let $E \subseteq \mathbb{R}^n$ and μ be a metric outer measure such that $0 < \mu(E) < \infty$. Relative to some oracle A, the function $p_{\mu,E}$ is a lower semi-computable discrete semimeasure.*

In order to connect the existence of such an outer measure μ to the existence of optimal oracles, we need to relate the semimeasure p_μ and Kolmogorov complexity. We achieve this using a fundamental result in algorithmic information theory.

Levin's optimal lower semicomputable subprobability measure, relative to an oracle A, on the dyadic rationals $\mathbb{D}$ is defined by

$$\mathbf{m}^A(d) = \sum_{\pi \,:\, U^A(\pi) = d} 2^{-|\pi|}.$$

▶ **Lemma 10.** *Let $E \subseteq \mathbb{R}^n$ and μ be a metric outer measure such that $0 < \mu(E) < \infty$. Let A be an oracle relative to which $p_{\mu,E}$ is lower semi-computable. Then is a constant $\alpha > 0$ such that $\mathbf{m}^A(d) \geq \alpha p_{\mu,E}(d)$, for every $d \in \mathbb{D}^n$.*

Proof. Case and Lutz [3], generalizing Levin's coding theorem [9, 10], showed that there is a constant c such that

$$\mathbf{m}^A(d_{r,i}) \leq 2^{-K^A(d_{r,i}) + K^A(r) + c},$$

for every $r \in \mathbb{N}$ and $d_{r,i} \in \mathbb{D}^n$. The optimality of $\mathbf{m}^A$ ensures that, for every lower semicomputable (relative to A) discrete semimeasure ν on $\mathbb{D}^n$,

$$\mathbf{m}^A(d_{r,i}) \geq \alpha\nu(d_{r,i}).$$ ◀

The results of this section have dealt with the dyadic rationals. However, we ultimately deal with the Kolmogorov complexity of Euclidean points. A result of Case and Lutz [3] relates the Kolmogorov complexity of Euclidean points with the complexity of dyadic rationals.

▶ **Lemma 11** ([3])**.** *Let $x \in [0,1)^n$, $A \subseteq \mathbb{N}$, and $r \in \mathbb{N}$. Let $Q_{r,i}$ be the (unique) dyadic cube at precision r containing x. Then*

$$K^A_r(x) = K^A(d_{r,i}) - O(\log r).$$

▶ **Lemma 12.** *Let $E \subseteq \mathbb{R}^n$ and μ be a metric outer measure such that $0 < \mu(E) < \infty$. Let A be an oracle relative to which $p_{\mu,E}$ is lower semi-computable. Then, for every oracle $B \subseteq \mathbb{N}$ and every $\epsilon > 0$, the set*

$$N = \{x \in E \mid (\exists^\infty)\, K^{A,B}_r(x) < K^A_r(x) - \epsilon r\}$$

has μ-measure zero.

We now have the machinery in place to prove the main theorem of this section.

▶ **Theorem 13.** *Let $E \subseteq \mathbb{R}^n$ with $\dim_H(E) = s$. Suppose there is a metric outer measure μ such that*

$$0 < \mu(E) < \infty,$$

and either

1. *$\mu \ll \mathcal{H}^{s-\delta}$, for every $\delta > 0$, or*
2. *$\mathcal{H}^s \ll \mu$ and $\mathcal{H}^s(E) > 0$.*

Then E has an optimal Hausdorff oracle A.

Proof. Let $A \subseteq \mathbb{N}$ be a Hausdorff oracle for E such that $p_{\mu,E}$ is computable relative to A. Note that such an oracle exists by the point-to-set principle and routine encoding. We will show that A is optimal for E.

For the sake of contradiction, suppose that there is an oracle B and $\epsilon > 0$ such that, for every $x \in E$ either

1. $\dim^{A,B}(x) < s - \epsilon$, or
2. there are infinitely many r such that $K^{A,B}_r(x) < K^A_r(x) - \epsilon r$.

Let N be the set of all x for which the second item holds. By Lemma 12, $\mu(N) = 0$. We also note that, by the point-to-set principle,

$$\dim_H(E - N) \leq s - \epsilon,$$

and so $\mathcal{H}^s(E - N) = 0$.

To achieve the desired contradiction, we first assume that $\mu \ll \mathcal{H}^{s-\delta}$, for every $\delta > 0$. Since $\mu \ll \mathcal{H}^{s-\delta}$, and $\dim_H(E - N) < s - \epsilon$,

$$\mu(E - N) = 0.$$

Since μ is a metric outer measure,

$$\begin{aligned} 0 &< \mu(E) \\ &\leq \mu(N) + \mu(E - N) \\ &= 0, \end{aligned}$$

a contradiction.

Now suppose that $\mathcal{H}^s \ll \mu$ and $\mathcal{H}^s(E) > 0$. Then, since $\mathcal{H}^s$ is an outer measure, $\mathcal{H}^s(E) > 0$ and $\mathcal{H}^s(E - N) = 0$ we must have $\mathcal{H}^s(N) > 0$. However this implies that $\mu(N) > 0$, and we again have the desired contradiction. Thus A is an optimal Hausdorff oracle for E and the proof is complete. ◀

Recall that $E \subseteq [0,1)^n$ is called an s-set if

$$0 < \mathcal{H}^s(E) < \infty.$$

Since $\mathcal{H}^s$ is a metric outer measure, and trivially absolutely continuous with respect to itself, we have the following corollary.

▶ **Corollary 14.** *Let $E \subseteq [0,1)^n$ be an s-set. Then there is an optimal Hausdorff oracle for E.*

3.2 Sets with optimal Hausdorff oracles

We now show that every analytic set has optimal Hausdorff oracles.

▶ **Lemma 15.** *Every analytic set E has optimal Hausdorff oracles.*

Proof. We begin by assuming that E is compact, and let $s = \dim_H(E)$. Then for every $t < s$, $\mathcal{H}^t(E) > 0$. Thus, by Theorem 1(1), there is a sequence of compact subsets $F_1, F_2, \ldots$ of E such that

$$\dim_H(\bigcup_n F_n) = \dim_H(E),$$

and, for each n,

$$0 < \mathcal{H}^{s_n}(F_n) < \infty,$$

where $s_n = s - 1/n$. Therefore, by Theorem 13, each set F_n has optimal Hausdorff oracles. Hence, by Proposition 7, E has optimal Hausdorff oracles and the conclusion follows.

We now show that every Σ^0_2 set has optimal Hausdorff oracles. Suppose $E = \cup_n F_n$ is Σ^0_1, where each F_n is compact. As we have just seen, each F_n has optimal Hausdorff oracles. Therefore, by Proposition 7, E has optimal Hausdorff oracles and the conclusion follows.

Finally, let E be analytic. By Theorem 1(2), there is a Σ^0_2 subset F of the same Hausdorff dimension as E. We have just seen that F must have an optimal Hausdorff oracle. Since $\dim_H(F) = \dim_H(E)$, by Observation 6 E has optimal Hausdorff oracles, and the proof is complete. ◀

Crone, Fishman and Jackson [4] have recently shown that, assuming the Axiom of Determinacy (AD)[5], *every* subset E has a Borel subset F such that $\dim_H(F) = \dim_H(E)$. This, combined with Lemma 15, yields the following corollary.

▶ **Corollary 16.** *Assuming AD, every set $E \subseteq \mathbb{R}^n$ has optimal Hausdorff oracles.*

▶ **Lemma 17.** *Suppose that $E \subseteq \mathbb{R}^n$ satisfies $\dim_H(E) = \dim_P(E)$. Then E has an optimal Hausdorff oracle. Moreover, the join (A, B) is an optimal Hausdorff oracle, where A and B are Hausdorff and packing oracles, respectively, of E.*

[5] Note that AD is inconsistent with the axiom of choice.

Proof. Let A be a Hausdorff oracle for E and let B be a packing oracle for E. We claim that that the join (A, B) is an optimal Hausdorff oracle for E. By the point-to-set principle, and the fact that extra information cannot increase effective dimension,

$$\begin{aligned}\dim_H(E) &= \sup_{x\in E} \dim^A(x)\\ &\geq \sup_{x\in E} \dim^{A,B}(x)\\ &\geq \dim_H(E).\end{aligned}$$

Therefore

$$\dim_H(E) = \sup_{x\in E} \dim^{A,B}(x),$$

and the first condition of optimal Hausdorff oracles is satisfied.

Let $C \subseteq \mathbb{N}$ be an oracle and $\epsilon > 0$. By the point-to-set principle,

$$\dim_H(E) \leq \sup_{x\in E} \dim^{A,B,C}(x),$$

so there is an $x \in E$ such that

$$\dim_H(E) - \epsilon/4 < \dim^{A,B,C}(x).$$

Let r be sufficiently large. Then, by our choice of B and the fact that additional information cannot increase the complexity of a point,

$$\begin{aligned}K^{A,B}_r(x) &\leq K^B_r(x)\\ &\leq \dim_P(E)r + \epsilon r/4\\ &= \dim_H(E)r + \epsilon r/4\\ &< \dim^{A,B,C}(x)r + \epsilon r/2\\ &\leq K^{A,B,C}_r(x) + \epsilon r.\end{aligned}$$

Since the oracle C and ϵ were arbitrarily, the proof is complete. ◀

3.3 Sets without optimal Hausdorff oracles

In the previous section, we gave general conditions for a set E to have optimal Hausdorff oracles. Indeed, we saw that under the axiom of determinacy, every set has optimal Hausdorff oracles.

However, assuming the axiom of choice (AC) and the continuum hypothesis (CH), we are able to construct sets without optimal Hausdorff oracles.

▶ **Lemma 18.** *Assume AC and CH. Then, for every $s \in (0, 1)$, there is a subset $E \subseteq \mathbb{R}$ with $\dim_H(E) = s$ such that E does not have optimal Hausdorff oracles.*

Let $s \in (0, 1)$. We begin by defining two sequences of natural numbers, $\{a_n\}$ and $\{b_n\}$. Let $a_1 = 2$, and $b_1 = \lfloor 2/s \rfloor$. Inductively define $a_{n+1} = b_n^2$ and $b_{n+1} = \lfloor a_{n+1}/s \rfloor$. Note that

$$\lim_n a_n/b_n = s.$$

Using AC and CH, we order the subsets of the natural numbers such that every subset has countably many predecessors. For every countable ordinal α, let $f_\alpha : \mathbb{N} \to \{\beta \mid \beta < \alpha\}$ be a function such that each ordinal β strictly less than α is mapped to by infinitely many n. Note that such a function exists, since the range is countable assuming CH.

We will define real numbers x_α, y_α via transfinite induction. Let x_1 be a real which is random relative to A_1. Let y_1 be the real whose binary expansion is given by

$$y_1[r] = \begin{cases} 0 & \text{if } a_n < r \leq b_n \text{ for some } n \in \mathbb{N} \\ x_1[r] & \text{otherwise} \end{cases}$$

For the induction step, suppose we have defined our points up to α. Let x_α be a real number which is random relative to the join of $\bigcup_{\beta<\alpha}(A_\beta, x_\beta)$ and A_α. This is possible, as we are assuming that this union is countable. Let y_α be the point whose binary expansion is given by

$$y_\alpha[r] = \begin{cases} x_\beta[r] & \text{if } a_n < r \leq b_n, \text{ where } f_\alpha(n) = \beta \\ x_\alpha[r] & \text{otherwise} \end{cases}$$

Finally, we define our set $E = \{y_\alpha\}$. This set E satisfies $\dim_H(E) = s$, however E does not have an optimal Hausdorff oracle.

3.3.1 Generalization to higher dimension

In this section, we use Lemma 18 to show that there are sets without optimal Hausdorff oracles in $\mathbb{R}^n$ of every possible dimension. We will need the following lemma on giving sufficient conditions for a product set to have optimal Hausdorff oracles. Interestingly, we need the product formula to hold for arbitrary sets, first proven by Lutz [17]. Under the assumption that F is regular, the product formula gives

$$\dim_H(F \times G) = \dim_H(F) + \dim_H(G) = \dim_P(F) + \dim_H(G),$$

for every set G.

▶ **Lemma 19.** *Let $F \subseteq \mathbb{R}^n$ be a set such that $\dim_H(F) = \dim_P(F)$, let $G \subseteq \mathbb{R}^m$ and let $E = F \times G$. Then E has optimal Hausdorff oracles if and only if G has optimal Hausdorff oracles.*

▶ **Theorem 20.** *Assume AC and CH. Then for every $n \in \mathbb{N}$ and $s \in (0, n)$, there is a subset $E \subseteq \mathbb{R}^n$ with $\dim_H(E) = s$ such that E does not have optimal Hausdorff oracles.*

4 Marstrand's Projection Theorem

The following theorem, due to Lutz and Stull [20], gives sufficient conditions for strong lower bounds on the complexity of projected points.

▶ **Theorem 21.** *Let $z \in \mathbb{R}^2$, $\theta \in [0, \pi]$, $C \subseteq \mathbb{N}$, $\eta \in \mathbb{Q} \cap (0,1) \cap (0, \dim(z))$, $\varepsilon > 0$, and $r \in \mathbb{N}$. Assume the following are satisfied.*

1. *For every $s \leq r$, $K_s(\theta) \geq s - \log(s)$.*
2. *$K_r^{C,\theta}(z) \geq K_r(z) - \varepsilon r$.*

Then,

$$K_r^{C,\theta}(p_\theta z) \geq \eta r - \varepsilon r - \frac{4\varepsilon}{1-\eta} r - O(\log r)\,.$$

The second condition of this theorem requires the oracle (C, θ) to give essentially no information about z. The existence of optimal Hausdorff oracles gives a sufficient condition for this to be true, for all sufficiently large precisions. Thus we are able to show that Marstrands projection theorem holds for any set with optimal Hausdorff oracles.

▶ **Theorem 22.** *Suppose $E \subseteq \mathbb{R}^2$ has an optimal Hausdorff oracle. Then for almost every $\theta \in [0, \pi]$,*

$$\dim_H(p_\theta E) = \min\{\dim_H(E), 1\}.$$

This shows that Marstrand's theorem holds for every set E with $\dim_H(E) = s$ satisfying any of the following:

1. E is analytic.
2. $\dim_H(E) = \dim_P(E)$.
3. $\mu \ll \mathcal{H}^{s-\delta}$, for every $\delta > 0$ for some metric outer measure μ such that $0 < \mu(E) < \infty$.
4. $\mathcal{H}^s \ll \mu$ and $\mathcal{H}^s(E) > 0$, for some metric outer measure μ such that $0 < \mu(E) < \infty$.

For example, the existence of exact gauged Hausdorff measures on E guarantees the existence of optimal Hausdorff oracles.

▶ Example. Let E be a set with $\dim_H(E) = s$ and $\mathcal{H}^s(E) = 0$. Suppose that $0 < \mathcal{H}^\phi(E) < \infty$, where $\phi(t) = \frac{t^s}{\log \frac{1}{t}}$. Since $\mathcal{H}^\phi \ll \mathcal{H}^{s-\delta}$ for every $\delta > 0$, Theorem 13 implies that E has optimal Hausdorff oracles, and thus Marstrand's theorem holds for E.

▶ Example. Let E be a set with $\dim_H(E) = s$ and $\mathcal{H}^s(E) = \infty$. Suppose that $0 < \mathcal{H}^\phi(E) < \infty$, where $\phi(t) = t^s \log \frac{1}{t}$. Since $\mathcal{H}^s \ll \mathcal{H}^\phi$, Theorem 13 implies that E has optimal Hausdorff oracles, and thus Marstrand's theorem holds for E.

4.1 Counterexample to Marstrand's theorem

In this section we show that there are sets for which Marstrand's theorem does not hold. While not explicitly mentioning optimal Hausdorff oracles, the construction is very similar to the construction in Section 3.3.

▶ **Theorem 23.** *Assuming AC and CH, for every $s \in (0, 1)$ there is a set E such that $\dim_H(E) = 1 + s$ but*

$$\dim_H(p_\theta E) = s$$

for every $\theta \in (\pi/4, 3\pi/4)$.

This is a modest generalization of Davies' theorem to sets with Hausdorff dimension strictly greater than one. In the next section we give a new proof of Davies' theorem by generalizing this construction to the endpoint $s = 0$.

We will need the following simple observation.

▶ **Observation 24.** *Let $r \in \mathbb{N}$, $s \in (0, 1)$, and $\theta \in (\pi/8, 3\pi/8)$. Then for every dyadic rectangle*

$$R = [d_x - 2^{-r}, d_x + 2^{-r}] \times [d_y - 2^{-sr}, d_y + 2^{-sr}],$$

there is a point $z \in R$ such that $K^\theta_r(p_\theta z) \leq sr + o(r)$.

For every $r \in \mathbb{N}$, $\theta \in (\pi/4, 3\pi/4)$, binary string x of length r and string y of length sr, let $g_\theta(x, y) \mapsto z$ be a function such that

$$K^\theta_r(p_\theta\,(x, z)) \leq sr + o(r).$$

That is, g_θ, given a rectangle

$$R = [d_x - 2^{-r}, d_x + 2^{-r}] \times [d_y - 2^{-sr}, d_y + 2^{-sr}],$$

outputs a value z such that $K_r(p_\theta(x, z))$ is small.

Let $s \in (0,1)$. We begin by defining two sequences of natural numbers, $\{a_n\}$ and $\{b_n\}$. Let $a_1 = 2$, and $b_1 = \lfloor 2/s \rfloor$. Inductively define $a_{n+1} = b_n^2$ and $b_{n+1} = \lfloor a_{n+1}/s \rfloor$. We will also need, for every ordinal α, a function $f_\alpha : \mathbb{N} \to \{\beta \,|\, \beta < \alpha\}$ such that each ordinal $\beta < \alpha$ is mapped to by infinitely many n. Note that such a function exists, since the range is countable assuming CH.

Using AC and CH, we first order the subsets of the natural numbers and we order the angles $\theta \in (\pi/4, 3\pi/4)$ so that each has at most countably many predecessors.

We will define real numbers x_α, y_α and z_α inductively. Let x_1 be a real which is random relative to A_1. Let y_1 be a real which is random relative to (A_1, x_1). Define z_1 to be the real whose binary expansion is given by

$$z_1[r] = \begin{cases} g_{\theta_1}(x_1, y_1)[r] & \text{if } a_n < r \leq b_n \text{ for some } n \in \mathbb{N} \\ y_1[r] & \text{otherwise} \end{cases}$$

For the induction step, suppose we have defined our points up to ordinal α. Let x_α be a real number which is random relative to the join of $\bigcup_{\beta<\alpha}(A_\beta, x_\beta)$ and A_α. Let y_α be random relative to the join of $\bigcup_{\beta<\alpha}(A_\beta, x_\beta)$, A_α and x_α. This is possible, as we are assuming CH, and so this union is countable. Let z_α be the point whose binary expansion is given by

$$z_\alpha[r] = \begin{cases} g_{\theta_\beta}(x_\alpha, y_\alpha)[r] & \text{if } a_n < r \leq b_n, \text{ for } f_\alpha(n) = \beta \\ y_\alpha[r] & \text{otherwise} \end{cases}$$

Finally, we define our set $E = \{(x_\alpha, z_\alpha)\}$.

4.2 Generalization to the endpoint

▶ **Theorem 25.** *Assuming AC and CH, there is a set E such that $\dim_H(E) = 1$ but*

$$\dim_H(p_\theta E) = 0$$

for every $\theta \in (\pi/4, 3\pi/4)$.

For every $r \in \mathbb{N}$, $\theta \in (\pi/4, 3\pi/4)$, binary string x of length r and string y of length sr, let $g^s_\theta(x, y) \mapsto z$ be a function such that

$$K^\theta_r(p_\theta\,(x, z)) \leq sr + o(r).$$

That is, g^s_θ, given a rectangle

$$R = [d_x - 2^{-r}, d_x + 2^{-r}] \times [d_y - 2^{-sr}, d_y + 2^{-sr}],$$

outputs a value z such that $K_r(p_\theta(x, z))$ is small.

We begin by defining two sequences of natural numbers, $\{a_n\}$ and $\{b_n\}$. Let $a_1 = 2$, and $b_1 = 4$. Inductively define $a_{n+1} = b_n^2$ and $b_{n+1} = (n+1)\lfloor a_{n+1} \rfloor$. We will also need, for every ordinal α, a function $f_\alpha : \mathbb{N} \to \{\beta \,|\, \beta < \alpha\}$ such that each ordinal $\beta < \alpha$ is mapped to by infinitely many n. Note that such a function exists, since the range is countable assuming CH.

Using AC and CH, we first order the subsets of the natural numbers and we order the angles $\theta \in (\pi/4, 3\pi/4)$ so that each has at most countably many predecessors.

We will define real numbers x_α, y_α and z_α inductively. Let x_1 be a real which is random relative to A_1. Let y_1 be a real which is random relative to (A_1, x_1). Define z_1 to be the real whose binary expansion is given by

$$z_1[r] = \begin{cases} g^1_{\theta_1}(x_1, y_1)[r] & \text{if } a_n < r \leq b_n \text{ for some } n \in \mathbb{N} \\ y_1[r] & \text{otherwise} \end{cases}$$

For the induction step, suppose we have defined our points up to ordinal α. Let x_α be a real number which is random relative to the join of $\bigcup_{\beta<\alpha}(A_\beta, x_\beta)$ and A_α. Let y_α be random relative to the join of $\bigcup_{\beta<\alpha}(A_\beta, x_\beta)$, A_α and x_α. This is possible, as we are assuming CH, and so this union is countable. Let z_α be the point whose binary expansion is given by

$$z_\alpha[r] = \begin{cases} g^{1/n}_{\theta_\beta}(x_\alpha, y_\alpha)[r] & \text{if } a_n < r \leq b_n, \text{ for } f_\alpha(n) = \beta \\ y_\alpha[r] & \text{otherwise} \end{cases}$$

Finally, we define our set $E = \{(x_\alpha, z_\alpha)\}$.

5 Optimal Packing Oracles

Similarly, we can define optimal *packing* oracles for a set.

▶ **Definition 26.** *Let $E \subseteq \mathbb{R}^n$ and $A \subseteq \mathbb{N}$. We say that A is an optimal packing oracle (or packing optimal) for E if the following conditions are satisfied.*

1. *A is a packing oracle for E.*
2. *For every $B \subseteq \mathbb{N}$ and every $\epsilon > 0$ there is a point $x \in E$ such that $\mathrm{Dim}^{A,B}(x) \geq \dim_P(E) - \epsilon$ and for almost every $r \in \mathbb{N}$*
$$K^{A,B}_r(x) \geq K^A_r(x) - \epsilon r.$$

Let $E \subseteq \mathbb{R}^n$ and $A \subseteq \mathbb{N}$. For $B \subseteq \mathbb{N}$, $\epsilon > 0$ define the set

$$N(A, B, \epsilon) = \{x \in E \mid (\forall^\infty r)\, K^{A,B}_r(x) \geq K^A_r(x) - \epsilon r\}.$$

▶ **Proposition 27.** *Let $E \subseteq \mathbb{R}^n$ be a set such that $\dim_P(E) > 0$ and let A be an oracle. Then A is packing optimal for E if and only if A is a packing oracle and for every $B \subseteq \mathbb{N}$ and $\epsilon > 0$, $\dim_P(N(A, B, \epsilon)) = \dim_P(E)$.*

▶ **Lemma 28.** *Let $E \subseteq \mathbb{R}^n$. If A is packing optimal for E, then the join $C = (A, B)$ is packing optimal for E for every oracle B.*

We now give some basic closure properties of the class of sets with optimal packing oracles.

▶ **Observation 29.** *Let $F \subseteq E$. If $\dim_P(F) = \dim_P(E)$ and F has an optimal packing oracle, then E has an optimal packing oracle.*

We can also show that having optimal packing oracles is closed under countable unions.

▶ **Lemma 30.** *Let $E_1, E_2, \ldots$ be a countable sequence of sets and let $E = \cup_n E_n$. If every set E_n has an optimal packing oracle, then E has an optimal packing oracle.*

We will need a specific set which has optimal Hausdorff and optimal packing oracles. For every $0 \leq \alpha < \beta \leq 1$ define the set

$$D_{\alpha,\beta} = \{x \in (0,1) \mid \dim(x) = \alpha \text{ and } \mathrm{Dim}(x) = \beta\}.$$

▶ **Lemma 31.** *For every $0 \leq \alpha < \beta \leq 1$, $D_{\alpha,\beta}$ has optimal Hausdorff and optimal packing oracles and*

$$\dim_H(D_{\alpha,\beta}) = \alpha$$
$$\dim_P(D_{\alpha,\beta}) = \beta.$$

5.1 Sufficient conditions for optimal packing oracles

▶ **Lemma 32.** *Let $E \subseteq \mathbb{R}^n$ be a set such that $\dim_H(E) = \dim_P(E) = s$. Then E has optimal Hausdorff and optimal packing oracles.*

▶ **Theorem 33.** *Let $E \subseteq \mathbb{R}^n$ with $\dim_P(E) = s$. Suppose there is a metric outer measure μ such that*

$$0 < \mu(E) < \infty,$$

and either
1. *$\mu \ll \mathcal{P}^s$, or*
2. *$\mathcal{P}^s \ll \mu$ and $\mathcal{P}^s(E) > 0$.*

Then E has an optimal packing oracle A.

We now show that every analytic set has optimal packing oracles.

▶ **Lemma 34.** *Every analytic set E has optimal packing oracles.*

5.2 Sets without optimal oracles

In this section, we state results which show that, assuming CH and AC, there are sets without Hausdorff optimal and without packing optimal oracles of arbitrary dimension.

▶ **Theorem 35.** *Assuming CH and AC, for every $0 < s_1 < s_2 \leq 1$ there is a set $E \subseteq \mathbb{R}$ which does not have Hausdorff optimal nor packing optimal oracles such that*

$$\dim_H(E) = s_1 \text{ and } \dim_P(E) = s_2.$$

▶ **Corollary 36.** *Assuming CH and AC, for every $0 < s_1 < s_2 \leq 1$ there is a set $E \subseteq \mathbb{R}$ which has optimal Hausdorff oracles but does not have optimal packing oracles such that*

$$\dim_H(E) = s_1 \text{ and } \dim_P(E) = s_2.$$

▶ **Theorem 37.** *Assuming CH and AC, for every $0 < s_1 < s_2 \leq 1$ there is a set $E \subseteq \mathbb{R}$ which has optimal packing oracles but does not have optimal Hausdorff oracles such that*

$$\dim_H(E) = s_1 \text{ and } \dim_P(E) = s_2.$$

References

1 Krishna B. Athreya, John M. Hitchcock, Jack H. Lutz, and Elvira Mayordomo. Effective strong dimension in algorithmic information and computational complexity. *SIAM J. Comput.*, 37(3):671–705, 2007. doi:10.1137/S0097539703446912.

2 Christopher J. Bishop and Yuval Peres. *Fractals in probability and analysis*, volume 162 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2017. doi:10.1017/9781316460238.

3 Adam Case and Jack H. Lutz. Mutual dimension. *ACM Transactions on Computation Theory*, 7(3):12, 2015.

4 Logan Crone, Lior Fishman, and Stephen Jackson. Hausdorff dimension regularity properties and games. *arXiv preprint*, 2020. arXiv:2003.11578.

5 Roy O. Davies. Two counterexamples concerning Hausdorff dimensions of projections. *Colloq. Math.*, 42:53–58, 1979.

6 Rod Downey and Denis Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer-Verlag, 2010.

7 Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, third edition, 2014.

8 Kenneth Falconer, Jonathan Fraser, and Xiong Jin. Sixty years of fractal projections. In *Fractal geometry and stochastics V*, pages 3–25. Springer, 2015.

9 Leonid A. Levin. On the notion of a random sequence. *Soviet Math Dokl.*, 14(5):1413–1416, 1973.

10 Leonid Anatolevich Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974.

11 Ming Li and Paul M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, third edition, 2008.

12 Jack H. Lutz. Dimension in complexity classes. *SIAM J. Comput.*, 32(5):1236–1259, 2003.

13 Jack H. Lutz. The dimensions of individual strings and sequences. *Inf. Comput.*, 187(1):49–79, 2003.

14 Jack H. Lutz and Neil Lutz. Algorithmic information, plane Kakeya sets, and conditional dimension. *ACM Trans. Comput. Theory*, 10(2):Art. 7, 22, 2018. doi:10.1145/3201783.

15 Jack H Lutz and Neil Lutz. Who asked us? how the theory of computing answers questions about analysis. In *Complexity and Approximation*, pages 48–56. Springer, 2020.

16 Jack H. Lutz and Elvira Mayordomo. Dimensions of points in self-similar fractals. *SIAM J. Comput.*, 38(3):1080–1112, 2008.

17 Neil Lutz. Fractal intersections and products via algorithmic dimension. In *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, 2017.

18 Neil Lutz and D. M. Stull. Bounding the dimension of points on a line. In *Theory and applications of models of computation*, volume 10185 of *Lecture Notes in Comput. Sci.*, pages 425–439. Springer, Cham, 2017.

19 Neil Lutz and D. M. Stull. Dimension spectra of lines. In *Unveiling dynamics and complexity*, volume 10307 of *Lecture Notes in Comput. Sci.*, pages 304–314. Springer, Cham, 2017.

20 Neil Lutz and D. M. Stull. Projection theorems using effective dimension. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, 2018.

21 J. M. Marstrand. Some fundamental geometrical properties of plane sets of fractional dimensions. *Proc. London Math. Soc. (3)*, 4:257–302, 1954. doi:10.1112/plms/s3-4.1.257.

22 Pertti Mattila. Hausdorff dimension, orthogonal projections and intersections with planes. *Ann. Acad. Sci. Fenn. Ser. AI Math*, 1(2):227–244, 1975.

23 Pertti Mattila. *Geometry of sets and measures in Euclidean spaces: fractals and rectifiability*. Cambridge University Press, 1999.

24 Pertti Mattila. Hausdorff dimension, projections, and the fourier transform. *Publicacions matematiques*, pages 3–48, 2004.

25 Pertti Mattila. Hausdorff dimension, projections, intersections, and besicovitch sets. In *New Trends in Applied Harmonic Analysis, Volume 2*, pages 129–157. Springer, 2019.

26 Elvira Mayordomo. A Kolmogorov complexity characterization of constructive Hausdorff dimension. *Inf. Process. Lett.*, 84(1):1–3, 2002.

27 Elvira Mayordomo. Effective fractal dimension in algorithmic information theory. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms: Changing Conceptions of What is Computable*, pages 259–285. Springer New York, 2008.

28 Andre Nies. *Computability and Randomness.* Oxford University Press, Inc., New York, NY, USA, 2009.

29 Tuomas Orponen. Combinatorial proofs of two theorems of Lutz and Stull. *arXiv preprint*, 2020. `arXiv:2002.01743`.

30 D. M. Stull. Results on the dimension spectra of planar lines. In *43rd International Symposium on Mathematical Foundations of Computer Science*, volume 117 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 79, 15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

High Quality Consistent Digital Curved Rays via Vector Field Rounding

Takeshi Tokuyama
Department of Computer Science, School of Engineering,
Kwansei Gakuin University, Sanda, Japan

Ryo Yoshimura
Graduate School of Information Science and Technology, The University of Tokyo, Japan

Abstract

We consider the *consistent digital rays* (CDR) of curved rays, which approximates a set of curved rays emanating from the origin by the set of rooted paths (called *digital rays*) of a spanning tree of a grid graph. Previously, a construction algorithm of CDR for *diffused families* of curved rays to attain an $O(\sqrt{n \log n})$ bound for the distance between digital ray and the corresponding ray is known [11]. In this paper, we give a description of the problem as a rounding problem of the vector field generated from the ray family, and investigate the relation of the quality of CDR and the discrepancy of the range space generated from gradient curves of rays. Consequently, we show the existence of a CDR with an $O(\log^{1.5} n)$ distance bound for any diffused family of curved rays.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Computational Geometry, Discrepancy Theory, Consistent Digital Rays, Digital Geometry

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.58

Funding This work was supported by JSPS KAKENHI Grant Number 20H04143.

1 Introduction

Digital pictures and graphic displays are modeled by using a digital plane consisting of pixels in the square region $[0, n] \times [0, n]$. A *pixel* often means the unit square that is a cell of the integer grid, but it is represented by the grid point at its lower-left corner, and the unit square is called *pixel square* if necessary in this paper. In the digital plane, geometric objects are represented by sets of pixels. In such a pixel-based representation, geometric computation (e.g. the intersection computation) can be done pixel-wise using the pixel buffers equipped in GPU. Thus, the pixel-based representation of digital objects would lead to an additional methodology for geometric computation.

However, conversion of geometric objects into digital objects is a nontrivial problem [14], and it may cause several inconsistencies of computation. In particular, the digital objects representing basic objects in Euclidean geometry do not always satisfy Euclidean axioms. The first two Euclidean axioms are the properties on line segments: (1) we can draw a line segment between any given two points, and (2) we can extend a line segment straightly and continuously to a line. Also, it is implied that the line segment between two points is unique, and it is a subset of any longer line segment going through them. As a consequence, a nonempty intersection of two line segments must be either a point or a line segment (the second case happens if the line segments are on the same line). These axioms are also considered in non-Euclidean geometries, where line segments are replaced by geodesic curves.

A naive digital line segment representing the line segment pq between two pixels p and q is the set of pixels corresponding to pixel squares intersecting the real line segment pq. However, the axioms do not hold for this definition of digital line segments. As a consequence, as shown in Figure 1, the intersection of a pair of such digital line segments may have more

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 58; pp. 58:1–58:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

than one connected components in the 4-neighbor topology of the digital plane, which may cause inconsistency in computation. It is a curious and important issue in mathematics and computer science to investigate a digital representation of a family of geometric objects such that they satisfy discrete counterparts of the Euclidean axioms.

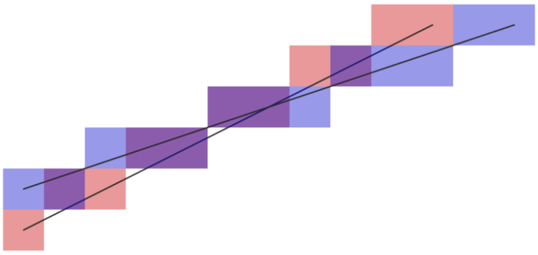


Figure 1 The intersection (purple pixel squares) of naive digital line segments may be disconnected.

The concept of *consistent digital rays* gives a model of digitization of a family of rays in the first quadrant[11, 12], which enables us to investigate the theoretical limit of digitization quantitatively by using the discrepancy theory [5, 16]. Here, a ray is a nondecreasing curve in the first quadrant emanating from the origin, and a pair of rays in the family do not intersect each other except at the origin (a concrete definition is given in Section 3).

Consider the triangular region Δ defined by $\{(x, y) \mid x \geq 0, y \geq 0, x+y \leq n\}$ in the plane, and the integer grid $G = \{(i, j) \mid i, j \in \{0, 1, \ldots, n\}, i + j \leq n\}$ in the region.

Each element of G is called a *pixel* (corresponding to the pixel in a digital picture). A pixel is called a *boundary pixel* if it lies on the off-diagonal boundary $x + y = n$ of Δ. The directed grid graph structure $\mathcal{G} = (G, E(\mathcal{G}))$ corresponding to the four-neighbor topology is given such that we have directed edges from $(i, j) \in G$ to $(i+1, j)$ and $(i, j+1)$ if $i+j \leq n-1$.

A *digital ray* is a directed path in $\mathcal{G}$ from the origin o to a pixel p. A digital ray is identified with the set of pixels on it, and regarded as a subset of G. Let us consider a family $\Pi = \{\Pi(p) \mid p \in G\}$ of digital rays. The family is called *consistent* if the following three properties hold:

1. *Uniqueness property*: For each $p \in G$, there exists a unique digital ray $\Pi(p)$ from the origin o to p in the family. We define $\Pi(o) = \{o\}$.
2. *Subsegment property*: If $q \in \Pi(p)$, then $\Pi(q) \subseteq \Pi(p)$.
3. *Prolongation property*: For each $\Pi(p)$, there is a (not necessarily unique) boundary pixel r such that $\Pi(p) \subseteq \Pi(r)$.

These properties are considered as the digital counterparts of the Euclidean axioms modified for the family of all halflines (called linear rays) emanating from the origin in the first quadrant.[1]

It is observed that the union of edge sets of paths in a consistent family of digital rays forms a (directed) spanning tree $\mathcal{T}$ of $\mathcal{G}$ rooted at o such that all leaves are boundary pixels (this condition corresponds to the prolongation property). The tree $\mathcal{T}$ is identified with the family Π of digital rays, and both of them are called CDR (*Consistent Digital Rays*). See the pictures (a) and (b) of Figure 2 for examples of CDR.

[1] The *shortest-path property* given in [12, 11, 7] is omitted by defining $\mathcal{G}$ as a directed graph in this paper.

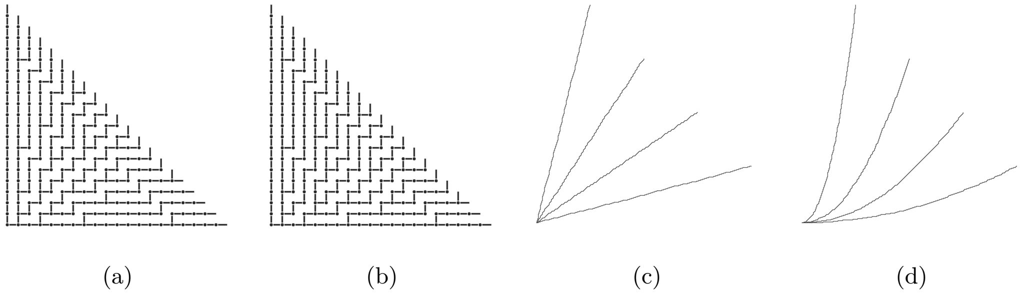


Figure 2 CDR for linear rays and parabolic rays in the triangular region of a 20 × 20 grid, and sampled linear and parabola digital rays in a 400 × 400 square grid.

Given a family of rays, it is desired to find a CDR approximating rays simultaneously. The quality of the approximation is measured by the largest distance between the digital ray $\mathit{\Pi}(p)$ and the corresponding ray $C(p)$ going through p over all $p \in G$. The Hausdorff distance is a popular distance between geometric objects, and considered in the previous works.

Historically, the theory started with how to realize digital straightness[14] to find a digitization of lines and line segments. Luby [15] first gave a construction of a CDR, where each $\mathit{\Pi}(p)$ simulates a linear ray within Hausdorff distance $O(\log n)$, and showed that the bound is asymptotically tight using geometric discrepancy. The construction was re-discovered by Chun et al. [12] in the formulation shown above. Christ et al. [10] gave a construction of consistent digital line segments where the lines need not go through the origin. There are works on variations and the high-dimensional generalizations [7, 8, 9].

The theory is extended by Chun et al. to families of curved rays [11]. A typical example is the family of parabolas $y = ax^2$ for $a \geq 0$. In Figure 2, the combinatorial difference between two CDRs (a) and (b) can be observed. The difference leads to the visual difference of digital rays illustrated in Figure 2, where it can be seen that the digital rays in (b) approximate parabolas as shown in (d) extended to a sufficiently large grid, while (a) approximates linear rays as shown in (c). A construction method of CDR for a wide class of families of curved rays called *diffused ray families* (its definition is given in Section 3.3) is given in [11]. However, the usage of discrepancy theory is limited because of difficulty to handle curved rays, and the attained distance bound is $O(\sqrt{n}\log n)$.

In this paper, we give a novel description of the problem as a rounding problem of a vector field, and regard the problem as a variant of the linear discrepancy problem. Intuitively, the rays are considered as geodesic curves for the vector field, and the *rounding* of the vector field naturally leads to a CDR. Then, in order to solve this variant of discrepancy problem, we apply the transference theory from the combinatorial discrepancy to the geometric discrepancy, and generate a tailor-made low-discrepancy pseudo-random sequence for the given family $\mathcal{F}$.

This enables us to prove the existence of a CDR with an $O(\log^{1.5} n)$ upper bound for the distance between rays and their corresponding digital rays for any diffused ray family. Although the above proof uses a non-constructive method in discrepancy theory, a CDR with a slightly weaker $O(\log^2 n)$ distance bound is computed in polynomial time.

2 Preliminaries on discrepancy theory

We introduce the definitions of three kinds of discrepancies used in this paper.

2.1 Range space and geometric Discrepancy

Consider a family $\mathcal{A}$ of subregions of $R = [0, n] \times [0, 1]$ and a set P of n points in R. The pair $(P, \mathcal{A})$ forms a *range space*. Let $\mathrm{vol}(A)$ be the area of $A \in \mathcal{A}$. We define

$$\begin{aligned} D(P, A) &= |\mathrm{vol}(A) - |P \cap A|| \quad \text{for} \quad A \in \mathcal{A}, \\ D(P, \mathcal{A}) &= \sup_{A \in \mathcal{A}} D(P, A), \;\text{ and} \\ D(n, \mathcal{A}) &= \inf_{|P|=n} D(P, \mathcal{A}). \end{aligned}$$

$D(P, \mathcal{A})$ and $D(n, \mathcal{A})$ are called the *geometric discrepancies* of the range space $(P, \mathcal{A})$ and the region family $\mathcal{A}$, respectively. See [16] for the geometric discrepancy theory. [2]

2.2 Combinatorial Discrepancy

For a finite set X, a family $\mathcal{S} \subseteq 2^X$ is called a *set system* on X. It generates a *hypergraph* $H = (X, \mathcal{S})$. A hypergraph coloring (bi-coloring) of H is a mapping $\chi : X \to \{-1, +1\}$, and we define $\chi(S) = \sum_{x \in S} \chi(x)$ for $S \in \mathcal{S}$. The *combinatorial discrepancy* is a measure of the balance of the coloring defined as follows:

$$\begin{aligned} \mathrm{disc}(\chi, \mathcal{S}) &= \max_{S \in \mathcal{S}} |\chi(S)|, \\ \mathrm{disc}(\mathcal{S}) &= \min_{\chi} \mathrm{disc}(\chi, \mathcal{S}). \end{aligned}$$

Given a range space $(P, \mathcal{A})$, $\mathcal{A}|_P = \{P \cap A \mid A \in \mathcal{A}\}$ is a set system on P, and we can consider its combinatorial discrepancy $\mathrm{disc}(\mathcal{A}|_P)$. We define the combinatorial discrepancy of the region family $\mathcal{A}$ by $\mathrm{disc}(n, \mathcal{A}) = \max_{|P|=n} \mathrm{disc}(\mathcal{A}|_P)$.

The combinatorial discrepancy of a range space and the geometric discrepancy are strongly related via *transference principle* (Theorem 14).

2.3 Linear discrepancy

Given a hypergraph $H = (X, \mathcal{S})$ and a real valued function $w : X \to [-1, 1]$ called *weight function*, we consider a function $\chi : X \to \{-1, 1\}$ called a *rounding* of w. For each $S \in \mathcal{S}$, $w(S)$ and $\chi(S)$ are the summations of the values of w and χ over S, respectively. The *linear discrepancy* of the rounding χ is

$$\mathrm{lindisc}(w, \chi) = \max_{S \in \mathcal{S}} |\chi(S) - w(S)|.$$

$\min_\chi \mathrm{lindisc}(w, \chi)$ and $\max_w \min_\chi \mathrm{lindisc}(w, \chi)$ are called the linear discrepancy of w and H, respectively. The combinatorial discrepancy $\mathrm{disc}(\mathcal{S})$ is equivalent to the linear discrepancy of the weight function $w \equiv 0$.

[2] The geometric discrepancy is defined more generally in [16] for range spaces in $[0, 1]^d$ instead of $[0, n] \times [0, 1]$.

3 Consistent Digital Rays

3.1 The structure of consistent digital rays

As mentioned in the introduction, a CDR is regarded as a rooted directed spanning tree $\mathcal{T}$ of the grid graph $\mathcal{G}$ on the triangular grid G, such that $\mathcal{T}$ has no leaf in the interior of Δ. Let $\ell(z)$ be the off-diagonal line defined by $x+y=z$. $L(k) = \{(x,y) \in G \mid x+y=k\} = \ell(k) \cap G$ is a *level set* of G for a natural number $k \leq n$. By definition, all leaves of $\mathcal{T}$ are in $L(n)$.

Each non-root pixel has exactly one incoming edge of $\mathcal{T}$. Also, as illustrated in Figure 3, there is a unique pixel (named branching pixel) in $L(k)$ with two outgoing edges for $k \neq n$, since $|L(k+1)| = |L(k)|+1$ and there is no leaf vertex in $L(k)$. Accordingly, there exists a point (not necessarily a pixel) $p \in \ell(k+1)$ such that all incoming edges to the pixels on the left (resp. right) of p are vertical (resp. horizontal). Such a point is called a *split point*, which partitions the incoming edges to each level into vertical and horizontal ones.

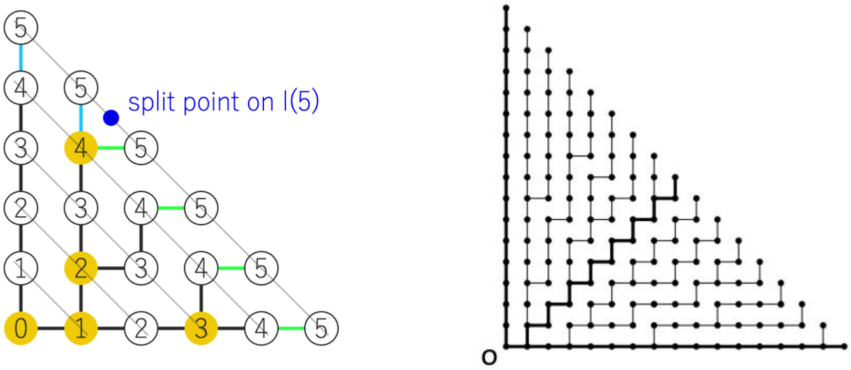


Figure 3 The branching pixels (colored yellow) and a split point are illustrated in the left picture, which shows the first five levels of the CDR in the right picture.

3.2 Off-diagonal distance between rays

A non-decreasing curve segment in Δ emanating from the origin is called a *partial ray*. We slightly abuse the notation so that a rooted path in $\mathcal{G}$ is also a partial ray, which consists of horizontal and vertical segments corresponding to its edges. We say that a partial ray terminates on $\ell(t)$ if it ends at a point on $\ell(t)$. A partial ray is called a *ray* if it terminates on the off-diagonal boundary $\ell(n)$ of Δ.

Given a partial ray C crossing $\ell(z)$, let $q_C(z) = (x_C(z), y_C(z))$ be the unique intersection point of C and $\ell(z)$. We define the discrete off-diagonal-wise L_∞ distance (*off-diagonal distance* in short) using $x_C(z)$ as follows:

Given partial rays C and C' both terminating on $\ell(m)$ for a natural number $m \leq n$, their off-diagonal distance is defined by

$$d_o(C, C') = \max_{k=1,2,\ldots,m} |x_C(k) - x_{C'}(k)|.$$

In other words, we measure the distance between two partial rays by the maximum horizontal distance (the vertical distance is the same) between their intersection points with $\ell(k)$ over natural numbers $k \leq m$. In particular, we can consider the off-diagonal distance $d_o(\Gamma, C)$ between a rooted path Γ in $\mathcal{G}$ and a partial ray C terminating at the same pixel.

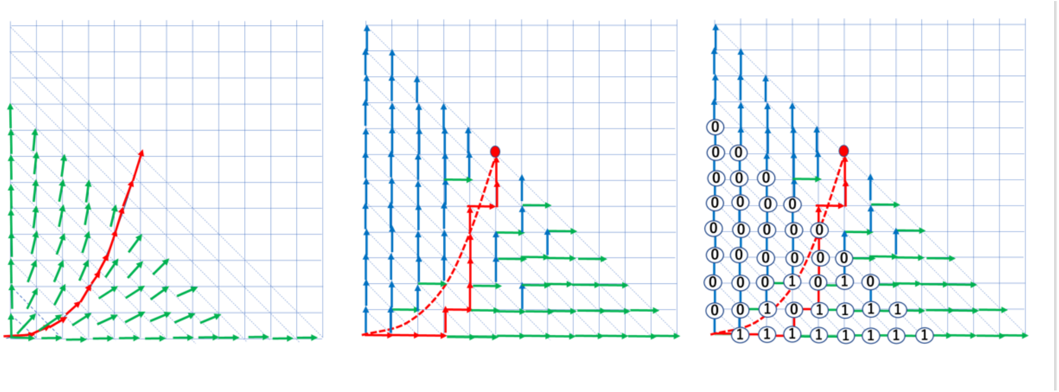


Figure 4 The vector field of the gradient vectors (left), a CDR approximating it (center), and its corresponding rounding χ (right, shown up to $L(8)$.)

The off-diagonal distance is a discrete variant of the L_∞-Hausdorff distance (i.e., the Hausdorff distance based on the L_∞ distance), which equals $\sup_{0<z\leq m} |x_C(z) - x_{C'}(z)|$ for partial rays. It is observed that the Hausdorff distance (i.e., the Hausdorff distance based on the Euclidean distance) between C and C' is at most $\sqrt{2}(d_o(C, C') + 1)$, and at least $d_o(C, C')$ (see textbooks or [12] for the definition of the Hausdorff distance). Thus, we use the off-diagonal distance in our analysis, since its asymptotic bound gives that of the Hausdorff distance.

3.3 CDR as rounding of a vector field

A family $\mathcal{F}$ of rays is called a *ray family* if for each point $p = (x, y) \in \Delta \setminus \{o\}$ there exists a unique ray $C(p)$ of $\mathcal{F}$ going through it. We denote the partial ray that is the part of $C(p)$ terminating at p by $\tilde{C}(p)$.

A ray family $\mathcal{F}$ is called *smooth* if each ray in $\mathcal{F}$ is differentiable.

Let us focus on a smooth ray family $\mathcal{F}$. We give a description of CDR as a *rounding* problem of a vector field induced from $\mathcal{F}$ to a discrete vector field on pixels (see Figure 4).

For $p = (x_p, y_p)$ for $x_p > 0$, suppose that the ray $C(p)$ is given by a function $y = f_p(x)$ in a neighbourhood of p. The slope of $C(p)$ at p is given by $f'_p(x_p)$ using the derivative of f. Since the slope is nonnegative, we can write $f'_p(x_p) = \frac{1-\alpha_p}{\alpha_p}$ uniquely by using a real number $0 < \alpha_p \leq 1$. It defines the *gradient vector* $\mathbf{V}_p = (\alpha_p, 1 - \alpha_p)$ to give the direction of the curve $C(p)$ at p normalized with respect to the L_1 norm. We set $\mathbf{V}_p = (0, 1)$ if $x_p = 0$ and $y_p > 0$. We do not define a gradient vector at $o = (0, 0)$. This defines a vector field $\mathbf{V} : \Delta \setminus \{o\} \to \mathbb{R}^2$ on the triangular region.[3]

As illustrated in the center picture of Figure 4, the CDR problem can be regarded as the problem to find an assignment of either $(1, 0)$ or $(0, 1)$ to each pixel of $G \setminus \{o\}$ such that the unit vector indicates the kind (horizontal or vertical) of the incoming edge of $\mathcal{T}$ to the pixel. If the CDR approximates the ray family $\mathcal{F}$, the assignment should approximate the vector field $\mathbf{V}$.

[3] If a potential function Φ to present gradient vectors as $(\frac{\partial \Phi}{\partial x}, \frac{\partial \Phi}{\partial y})$ is given, the rays are considered as geodesic paths in the potential field.

Each vector $\mathbf{V}_p$ is uniquely determined by $\alpha_p \in [0,1]$, and the vector field is converted to a $[0,1]$-valued function w defined by $w(p) = \alpha_p$. We call w the *gradient weight* of the vector field $\mathbf{V}$ in this paper. The vectors $(1,0)$ and $(0,1)$ are converted to 1 and 0 by this transformation.

Therefore, the CDR problem is converted to the problem to compute an assignment $\chi : G \setminus \{o\} \to \{0,1\}$ from the gradient weight (see the right picture of Figure 4). This is analogous to the linear discrepancy problem, if we scale the range of the weight from $[-1,1]$ to $[0,1]$. Thus, we call χ a *rounding* of w.

By definition, the off-diagonal distance between the digital ray $\Gamma = \Pi(p)$ and the partial ray $C = \tilde{C}(p)$ towards $p \in L(m)$ is $d_o(\Gamma, C) = \max_{k=1,2,\ldots m} |x_\Gamma(k) - x_C(k)|$, where $x_\Gamma(k)$ is the x-coordinate value of the pixel $q_\Gamma(k) = \Gamma \cap L(k)$. The following lemma relates the gradient weight and the rounding to the off-diagonal distance.

▶ **Lemma 1.** *$x_C(k) = \int_0^k w(q_C(z))dz$, and $x_\Gamma(k) = \sum_{i=1}^k \chi(q_\Gamma(i))$.*

Proof. If a ray goes through a point $q = (x,y)$ on $\ell(z)$ and reaches a point $(x+dx, y+dy)$ on $\ell(x+dz)$ for an infinitesimally small dz, then $dx = \alpha_q dz = w(q)dz$ by the definition of the gradient vector. If C is the ray, $q = q_C(z) = (x_C(z), y_C(z))$. Thus, $x_C(k) = \int_0^k w(q_C(z))dz$.

The x-value of a pixel $q = q_\Gamma(k)$ on a path Γ is the number of horizontal edges up to the pixel, which is the prefix sum of χ over the path Γ up to the level $L(k)$, and hence $x_\Gamma(k) = \sum_{i=1}^k \chi(q_\Gamma(i))$. ◀

A function f on Δ is called *off-diagonal monotone* if it is non-decreasing on each off-diagonal line $\ell(z)$. That is, $f(p) \geq f(q)$ if $x_p \geq x_q$ and $p,q \in \ell(z)$. It is called *strongly off-diagonal monotone* if it is increasing on each off-diagonal line.

The function χ corresponding to a spanning tree of G if $\chi(0,k) = 0$ and $\chi(k,0) = 1$ for $1 \leq k \leq n$ (i.e., the edges of $\mathcal{T}$ are vertical on the y-axis and horizontal on the x-axis). However, the spanning tree might have leaves in the interior of G (such a spanning tree is called a *weak CDR* in [7]). The spanning tree becomes a CDR if and only if χ is off-diagonal monotone, which is equivalent to the fact that there is a split point in each level.

We call a smooth ray family $\mathcal{F}$ *diffused* if the gradient weight w of its corresponding vector field is strongly off-diagonal monotone and continuous on each $\ell(z)$. This definition of the diffused ray family is equivalent to the one given in [11].

From now on, we focus on a CDR of a diffused family of rays, and regard it as the problem of seeking for a rounding χ minimizing the off-diagonal distance. The difference of this rounding problem from the ordinary linear discrepancy problem is as follows:

1. The set system is $\{\Pi(p) \mid p \in G\}$, which depends on $\mathcal{T}$, and hence on the choice of χ.
2. The rounding must preserve the off-diagonal monotonicity.
3. We must relate the off-diagonal distance to the discrepancy.

We apply the discrepancy theory to this vector field rounding problem.

4 Construction of CDR for diffused ray families

4.1 Construction algorithm of CDR via level-wise threshold rounding

We give a construction algorithm named *θ-threshold rounding algorithm* of a CDR approximating given diffused ray family $\mathcal{F}$ by using a $(0,1]$-valued sequence $\theta : \{1,2,\ldots n\} \to (0,1]$.

▶ **Definition 2.** *Given a gradient weight w and a $(0,1]$-valued sequence θ, the θ-threshold rounding χ of w is defined by the following:*

For $q \in L(k)$ $(k = 1,2,\ldots,n)$, $\chi(q) = 1$ if and only if $w(q) \geq \theta(k)$.

The construction algorithm is very simple: Given a diffused ray family $\mathcal{F}$, we consider its gradient weight w, compute its θ-threshold rounding, and obtain the corresponding CDR.

▶ **Example 3.** Consider the linear ray family $\mathcal{F} = \{C_a^{\mathrm{lin}} : y = ax \mid a \in [0, \infty]\}$, where C_∞^{lin} is the line $x = 0$. The derivative of $y = ax$ is a, which is equal to $\frac{y}{x}$. Hence, the slope of $C(p)$ at $p = (x, y)$ is $\frac{y}{x}$, and the vector field $\mathbf{V}$ is defined by $\mathbf{V}_p = (\frac{x}{x+y}, \frac{y}{x+y})$, and $w(p) = \frac{x}{x+y}$. If $p = (kt, k(1-t)) \in L(k)$, $w(p) = t$. Thus, $\chi(p) = 1$ if and only if $t \geq \theta(k)$.

▶ **Example 4.** Consider the parabola family $\mathcal{F} = \{C_a^{\mathrm{para}} : y = ax^2 \mid a \in [0, \infty]\}$, where C_∞^{para} is the line $x = 0$. The derivative of $y = ax^2$ is $y' = 2ax = \frac{2y}{x}$, and hence the slope of $C(p)$ at $p = (x, y)$ is $\frac{2y}{x}$. Thus, the vector field $\mathbf{V}$ is defined by $\mathbf{V}_p = (\frac{x}{x+2y}, \frac{2y}{x+2y})$, and $w(p) = \frac{x}{x+2y}$. If $p = (kt, k(1-t)) \in L(k)$, $w(p) = \frac{t}{2-t}$. Thus, $\chi(p) = 1$ if and only if $\frac{t}{2-t} \geq \theta(k)$.

The model of geometric computation to discuss the complexity and some more examples are given in the appendix.

4.2 Discrepancy that bounds the off-diagonal distance

The θ-threshold rounding algorithm is equivalent to the algorithm given in [11], where θ is fixed to be a random sequence or a known low-discrepancy sequence independently of choice of $\mathcal{F}$. In contrast to it, we seek for a tailor-made sequence θ to fit each ray family $\mathcal{F}$.

A ray $C \in \mathcal{F}$ defines its *gradient curve* $\varphi_C : \{(z, w(q_C(z))) \mid 0 < z \leq n\}$ in the (z, w) plane. Consider the family $\mathcal{F}^* = \{\varphi_C \mid C \in \mathcal{F}\}$ of gradient curves.

Given a curve $\varphi : w = f(z)$ in $\mathcal{F}^*$, let

$$\begin{aligned} R^-(\varphi, (a, b]) &= \{(z, w) \mid a < z \leq b,\ 0 \leq w < f(z)\} \text{ and} \\ R^+(\varphi, (a, b]) &= \{(z, w) \mid a < z \leq b,\ f(z) < w \leq 1\} \quad \text{for} \quad 0 \leq a \leq b \leq n. \end{aligned}$$

In other words, $R^-(\varphi, (a, b])$ (resp. $R^+(\varphi, (a, b])$) is the subregion of $(0, n] \times [0, 1]$ below (resp. above) φ and bounded by two vertical lines $z = a$ and $z = b$. We define the family of regions

$$\mathcal{A}_{\mathcal{F}^*} = \{R^\epsilon(\varphi, (a, b]) \mid 0 \leq a \leq b \leq n,\ \varphi \in \mathcal{F}^*,\ \epsilon \in \{+, -\}\} \cup \{(a, b] \times [0, 1] \mid 0 \leq a < b \leq n\}.$$

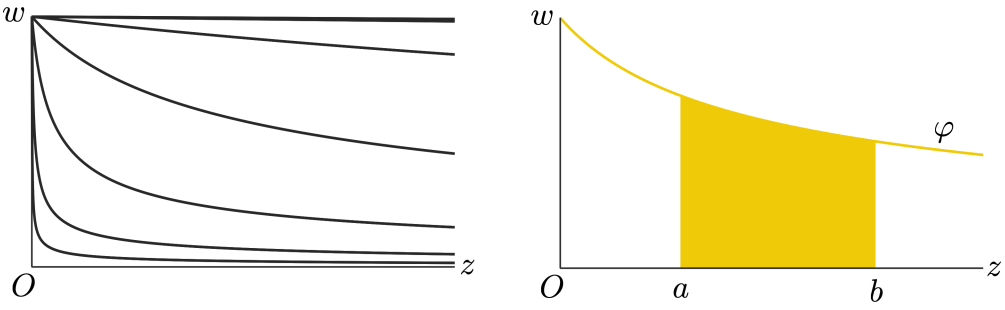


Figure 5 The gradient curves of parabola rays and a region $R^-(\varphi, (a, b])$ in $\mathcal{A}_{\mathcal{F}^*}$.

▶ **Example 5.** For the linear ray $C : y = ax$ $(a \geq 0)$, $w(q_c(z)) = \frac{1}{1+a}$, and hence φ_C is the horizontal line defined by $w = \frac{1}{1+a}$. Thus, $\mathcal{A}_{\mathcal{F}^*}$ is the family of axis parallel rectangles.

▶ **Example 6.** For the parabola ray $C : y = ax^2$ $(a \geq 0)$, $q_C(z) = (\frac{-1+\sqrt{1+4az}}{2a}, z - \frac{-1+\sqrt{1+4az}}{2a})$, and $w(q_C(z)) = \frac{1}{\sqrt{1+4az}}$. The curve φ_C is defined by $w = \frac{1}{\sqrt{1+4az}}$. The gradient curves and a region in $\mathcal{A}_{\mathcal{F}^*}$ for the family $\mathcal{F}$ of parabola rays are illustrated in Figure 5.

Let us fix the $(0,1]$-valued sequence θ, and focus on the rounding χ and corresponding CDR Π constructed by the θ-threshold rounding algorithm.

The point set $S(\theta) = \{s_i = (i, \theta(i)) \mid 1 \le i \le n\}$ is called the θ-*Hammersley point set*, or Hammersley point set if θ is implicitly given[4].

The following lemma shows a relation between the positions of points of $S(\theta)$ in the arrangement of gradient curves and the assignment of χ-values of pixels in the arrangement of rays.

▶ **Lemma 7.** *If $s_k = (k, \theta(k)) \in S(\theta)$ is below (resp. above) the gradient curve φ_C, $\chi(p) = 1$ (resp. $\chi(p) = 0$) for all pixels $p \in L(k)$ lying on the right (resp. left) of C.*

Proof. We assume s_k is below φ_C (the other case is analogous). Hence, $\theta(k) < w(q_C(k))$.

Because of the continuity and strong monotonicity of w on $\ell(k)$, there is a unique point $u \in \ell(k)$ satisfying $w(u) = \theta(k)$. The point u becomes a split point because of the definition of the θ-threshold rounding.

By the assumption, $w(u) = \theta(k) < w(q_C(k))$, and the strong monotonicity of w implies that u is on the left of C. Thus, each pixel $p \in L(k)$ on the right of C is also on the right of u, and thus $\chi(p) = 1$ because of the definition of the split points. ◀

We consider the geometric discrepancy $D(S(\theta), \mathcal{A}_{\mathcal{F}^*})$, and the following theorem tells the explicit relation of the discrepancy and the off-diagonal discrepancy.

▶ **Theorem 8.** *Suppose that $D(S(\theta), \mathcal{A}_{\mathcal{F}^*}) \le \delta(n)$ for a function δ. Then, $d_o(\Pi(p), \tilde{C}(p)) \le \delta(n) + 1$ for each pixel p in G.*

Proof. Let $S = S(\theta)$, $\Gamma = \Pi(p)$, and $C = \tilde{C}(p)$. Without loss of generality, we assume $p \in L(n)$. We assume the off-diagonal distance $\max_{1 \le k \le n} |x_\Gamma(k) - x_C(k)|$ between Γ and C is d, and derive $d \le \delta(n) + 1$ to prove the theorem.

From the assumption, there exists k_0 such that $|x_\Gamma(k_0) - x_C(k_0)| = d$. Thus, either $x_\Gamma(k_0) = x_C(k_0) + d$ or $x_\Gamma(k_0) = x_C(k_0) - d$, and we focus on the former case, since the latter case can be handled analogously.

Consider the first index $m > k_0$ such that $x_\Gamma(m) \le x_C(m)$. In other words, m is the first index after k_0 such that the pixel of Γ in the level $L(m)$ comes on the left of (or on) C. Such m exists because both Γ and C reach p. Thus,

$$x_\Gamma(m) - x_C(m) \le 0 = x_\Gamma(k_0) - x_C(k_0) - d \tag{1}$$

Consider $R = R^-(\varphi_C, (k_0, m-1]) \in \mathcal{A}_{\mathcal{F}^*}$, which is the region below φ_C and $k_0 < z \le m-1$. Let $N(S, R)$ be the number of points of S in R.

The path Γ is on the right of C in the range $k_0 \le z \le m-1$, and it is derived from Lemma 7 that $\chi(q_\Gamma(k)) = 1$ if $s_k \in R$. Thus, we have the following:

$$\sum_{k=k_0+1}^{m-1} \chi(q_\Gamma(k)) \ge \sum_{k:s_k \in R} \chi(q_\Gamma(k)) = \sum_{k:s_k \in R} 1 = N(S, R). \tag{2}$$

From Lemma 1, $x_\Gamma(j) = \sum_{k=1}^{j} \chi(q_\Gamma(k))$, and hence combined with (2),

$$x_\Gamma(m-1) - x_\Gamma(k_0) = \sum_{k=k_0+1}^{m-1} \chi(q_\Gamma(k)) \ge N(S, R). \tag{3}$$

[4] The original 2-dimensional Hammersley point set uses the van der Corput sequence as θ, but the notation is abused to allow to use a general θ.

By the definitions of R and φ_C,

$$\mathrm{vol}(R) = \int_{k_0}^{m-1} \varphi_C(z)dz = \int_{k_0}^{m-1} w(q_C(z))dz.$$

On the other hand, from Lemma 1,

$$\int_{k_0}^{m-1} w(q_C(z))dz = x_C(m-1) - x_C(k_0).$$

Thus,

$$\mathrm{vol}(R) = x_C(m-1) - x_C(k_0).$$

Since the geometric discrepancy $D(S, \mathcal{A}_{\mathcal{F}^*})$ is bounded by $\delta(n)$,

$$N(S,R) \quad \geq \quad \mathrm{vol}(R) - \delta(n) = x_C(m-1) - x_C(k_0) - \delta(n).$$

Thus, combined with (3),

$$x_\Gamma(m-1) - x_\Gamma(k_0) \geq N(S,R) \geq x_C(m-1) - x_C(k_0) - \delta(n),$$

and hence

$$x_\Gamma(m-1) - x_C(m-1) + \delta(n) \geq x_\Gamma(k_0) - x_C(k_0). \tag{4}$$

From (1) and (4), we have

$$x_\Gamma(m-1) - x_C(m-1) + \delta(n) \geq x_\Gamma(m) - x_C(m) + d.$$

Equivalently,

$$x_C(m) - x_C(m-1) + \delta(n) \geq x_\Gamma(m) - x_\Gamma(m-1) + d.$$

Since the x-value of a ray increases by at most one if the ray proceeds one level, $x_C(m) - x_C(m-1) \leq 1$ and $x_\Gamma(m) - x_\Gamma(m-1) \geq 0$. Hence, we have

$$1 + \delta(n) \geq d.$$

This is what we desire to obtain. ◀

5 Construction of the tailor-made low-discrepancy sequence

We give an upper bound of $D(n, \mathcal{A}_{\mathcal{F}^*})$ using the transference principle that derives an upper bound of the geometric discrepancy from that of the combinatorial discrepancy. Then, we construct θ such that $S(\theta) = \{(i, \theta(i)) \mid i = 1, 2, \ldots, n\}$ attains this discrepancy asymptotically.

5.1 Combinatorial property of the range space of gradient curves

▶ **Lemma 9.** *Given a diffused family $\mathcal{F}$, for any point $v = (z_0, w_0)$ in the rectangle $(0, n] \times [0, 1]$, there exists a unique gradient curve φ_C going through v.*

Proof. The range of w on $\ell(z_0)$ is $[0, 1]$ since $\mathcal{F}$ contains x-axis and y-axis. Because of the strong off-diagonal monotonicity and the continuity of w, there exists a point $q \in \ell(z_0)$ such that $w(q) = w_0$. Because of the definition of a ray family, there exists a unique ray $C \in \mathcal{F}$ going through q, and $w_C(q) = w(q)$. Thus, φ_C is the unique gradient curve going through v. ◀

▶ **Corollary 10.** *For a diffused family $\mathcal{F}$, each pair of gradient curves in $\mathcal{F}^*$ do not intersect each other in the domain $0 < z \leq n$.*

▶ **Definition 11** (Pseudo-rectangles). *Given a family $\mathcal{C}$ of x-monotone curves in $(0, n] \times [0, 1]$ such that each pair of curves do not intersect each other, a region bounded by a pair of curves and two vertical lines is called a pseudo-rectangle associated with $\mathcal{C}$. A (possibly infinite) set of such pseudo-rectangles is called a family of pseudo-rectangles associated with $\mathcal{C}$*

The following lemma follows the definition of $\mathcal{A}_{\mathcal{F}^*}$, Definition 11, and Corollary 10. See Figure 5 to get intuition.

▶ **Lemma 12.** *For a diffused ray family $\mathcal{F}$, $\mathcal{A}_{\mathcal{F}^*}$ is a family of pseudo-rectangles associated with $\mathcal{F}^*$.*

5.2 Discrepancies for the pseudo-rectangles

The Hammersley point set using the van der Corput sequence (van der Corput-Hammersley point set) is known to give an $O(\log n)$ bound for the geometric discrepancy for the family of axis-parallel rectangles (see [16]). However, it is known that its discrepancy becomes $\Omega(\sqrt{n})$ if we consider a rotated rectangle (Exercise 3, Section 2.1 of [16]), and hence the $O(\log n)$ bound cannot be applied to pseudo-rectangles. It seems difficult to directly convert the $O(\log n)$ bound of geometric discrepancy for rectangles to the one for pseudo-rectangles.

Fortunately, the combinatorial structure for the hypergraph of the range space of the pseudo-rectangles is the same as that of axis-parallel rectangles.

The problem to investigate the combinatorial discrepancy $\mathrm{disc}(n, \mathcal{R})$ for the family $\mathcal{R}$ of axis-parallel rectangles is called *Tusnády's problem*. An $O(\log^4 n)$ bound [4] was given by Beck, and it was improved by Bohus to $O(\log^3 n)$ as an application of *k-permutation problem* [6]. The current best bound is $O(\log^{1.5} n)$ given by Nikolov [17], although it is not constructive. The construction given by Bansal and Garg [2, 3] has an $O(\log^2 n)$ discrepancy, and their algorithm runs in polynomial time using the semi-definite programming as a subroutine.

Because the combinatorial discrepancy only depends on the combinatorial properties of the range space, all these bounds hold for the combinatorial discrepancy of a range space of pseudo-rectangles. Thus, we obtain the following theorem from Lemma 12.

▶ **Theorem 13.** *$disc(n, \mathcal{A}_{\mathcal{F}^*}) = O(\log^{1.5} n)$, and a set P of n points attaining $disc(\mathcal{A}_{\mathcal{F}^*}|_P) = O(\log^2 n)$ can be computed in polynomial time.*

It is known that an upper bound of the combinatorial discrepancy for range spaces can be converted to that of the geometric discrepancy as shown in the following theorem named *Transference Principle* or *Transference Lemma* (Proposition 1.8 of [16]):

▶ **Theorem 14** (Transference Principle). *Let $\mathcal{A}$ be a range space. If $D(n, \mathcal{A}) = o(n)$ and* $\mathrm{disc}(n, \mathcal{A}) = O(f(n))$ *for a function satisfying $f(2n) \leq (2 - \delta)f(n)$ for all n and fixed $\delta > 0$, then $D(n, \mathcal{A}) = O(f(n))$.*

The assumptions on $f(n)$ and the condition that $D(n, \mathcal{A}) = o(n)$ hold for the range space of pseudo-rectangles. Therefore, an upper bound of the combinatorial discrepancy is transferred to that of geometric discrepancy for the pseudo-rectangles. (A more general result is given by Aistleitner, Bilyk and Nikolov [1].) The transference is given in a constructive fashion such that a point set P giving the geometric discrepancy bound can be obtained in polynomial time in n if the coloring attaining the combinatorial discrepancy can be done in polynomial time. Thus, we have the following:

▶ **Theorem 15.** *$D(n, \mathcal{A}_{\mathcal{F}^*}) = O(\log^{1.5} n)$, and a set P of n points attaining $D(P, \mathcal{A}_{\mathcal{F}^*}) = O(\log^2 n)$ can be computed in polynomial time.*

Note: After the submission of this paper, Dutta [13] claimed an improved $O(\log^{7/4} n)$ combinatorial discrepancy for the Tusnády's problem with polynomial time construction. Accordingly, the corresponding $O(\log^2 n)$ bounds in Theorem 13, Theorem 15 and Theorem 17 is improved to $O(\log^{7/4} n)$ once the claim is confirmed.

5.3 Arraying a point set to obtain a uniform number sequence

We have shown that there exists a point set in $[0, n] \times [0, 1]$ attaining the $O(\log^{1.5} n)$ geometric discrepancy for the region family $\mathcal{A}_{\mathcal{F}^*}$. However, we need $\theta(i) \in [0, 1]$ such that its Hammersley point set $S(\theta) = \{s_i = (i, \theta(i)) \mid 1 \leq i \leq n\}$ forms a low-discrepancy point set to attain the discrepancy bound. We claim that any low-discrepancy point set for $\mathcal{A}_{\mathcal{F}^*}$ can be arrayed to become a Hammersley point set without losing the low-discrepancy property.

▶ **Lemma 16** (Arraying lemma). *If we have a set P of n points with $\delta(n)$ geometric discrepancy for $\mathcal{A}_{\mathcal{F}^*}$, we can construct a Hammersley point set P' with $O(\delta(n))$ geometric discrepancy.*

Proof. Consider the sorted list $p_1, p_2, \ldots, p_n$ of P in the abscissas in the (z, w) plane. Let C_i be the unique gradient curve in $\mathcal{F}^*$ going through p_i, and p'_i be the point on C_i with the abscissa i. In other words, each point p_i is moved along C_i to the position of the abscissa i. Now, we have the point set P'. Consider a region R bounded by a gradient curve $C \in \mathcal{F}^*$ and two vertical lines. Since each point is moved along a curve and no pair of curves intersect, a point p'_i is below C if and only if p_i is below C.

Consider the numbers $N(P, \ell)$ and $N(P', \ell)$ of points in P and P' to the left of a vertical line $\ell : z = a$, respectively. Since the points of P' are arrayed, $N(P', \ell) = \lfloor a \rfloor$. Since $D(P, \mathcal{A}_{\mathcal{F}^*}) \leq \delta(n)$ and $(0, a] \times [0, 1] \in \mathcal{A}_{\mathcal{F}^*}$ has the area a, $|N(P, \ell) - a| \leq \delta(n)$. Thus, $|N(P, \ell) - N(P', \ell)| \leq \delta(n) + 1$. Therefore, at most $\delta(n) + 1$ points of P move crossing ℓ, since the move of points keeps the sorting order. Thus, at most $2(\delta(n) + 1)$ points move crossing two vertical boundaries of R. Therefore, the discrepancy of P' is at most $3\delta(n) + 2 = O(\delta(n))$. Given P', the sequence θ such that $P' = S(\theta)$ is automatically obtained. ◀

Thus, θ is constructed as desired, and we obtain our main result shown below. Note that the asymptotic distance bounds hold for both of the off-diagonal and Hausdorff distances.

▶ **Theorem 17.** *For a diffused ray family $\mathcal{F}$, there exists a CDR with an $O(\log^{1.5} n)$ distance bound between a partial ray towards a pixel and its digital ray. A CDR with an $O(\log^2 n)$ distance bound can be computed in polynomial time in n.*

Proof. Immediate from Theorem 8, Theorem 15 and Lemma 16. ◀

6 Digital pseudoline arrangement

A family of curves is called a *pseudoline arrangement* if each pair of curves intersect at most once to each other. The consistent digital pseudoline arrangement is defined by Chun et al. [11].

One important class of the consistent digital pseudoline arrangement is given as a union of translated copies of a CDR $\mathcal{T}$. A translated copy $\mathcal{T}(s)$ is obtained by translating $\mathcal{T}$ so that the origin is translated to $(s, -s)$ for an integer s.

The union $\cup_{-k\leq s\leq k}\mathcal{T}(s)$ represents the set of digital rays emanating from $2k+1$ grid points on the off-diagonal line $x+y=0$. The union is called a family of *shifted digital rays*.

▶ **Example 18.** If we consider shifted digital rays using the CDR of linear rays given in Example 3, we can generate digital line segments for a line segments (with nonnegative slopes) between pixels in G as segments of shifted linear rays. This is a different construction of digital line segments from [10].

▶ **Example 19.** If we consider shifted digital rays using the CDR for parabola rays given in Example 4, we have an approximation of the family of parabolas with the vertical axes and peaks on the off-diagonal line $x+y=0$.

We can immediately apply our construction to improve the distance bound of shifted digital rays for a general diffused ray family to $O(\log^{1.5} n)$.

Another class of consistent digital pseudoline arrangements discussed in [11] is the digitized homogeneous polynomial family approximating the family $\{C_{j,a} \mid y = ax^j$ for $a > 0$ and $j \in \{1, 2, \ldots, k\}\}$ for an integer k. We can apply our formulation to construct a union of CDR for it, but unfortunately, we have technical difficulty to generalize the Arraying Lemma (Lemma 16) to guarantee an improved distance bound.

7 Concluding remarks

The distance bound $O(\log^{1.5} n)$ is near to the known $\Omega(\log n)$ lower bound, but it is curious whether we can improve it to $O(\log n)$. Moreover, if we remove the off-diagonal monotonicity condition on χ, we have a weak CDR. It is known that the distance bound for a weak CDR is reduced to $O(1)$ for the family of linear rays [7]. It is curious to investigate the weak CDR for a general ray family.

Developing a practical algorithm for computing theoretically guaranteed CDR is also an important problem. Although the θ-threshold rounding algorithm is very simple, the sequence θ attaining the $O(\log^{1.5} n)$ distance bound is not constructed explicitly. The one with $O(\log^2 n)$ distance bound has a polynomial time construction. However, we need to deal with hypergraphs on vertex sets with nearly n^2 vertices and polylogarithmic vertex degrees if we apply the transference principle. Moreover, the coloring of the hypergraph to attain the combinatorial discrepancy in [2, 3] uses the semi-definite programming(SDP). Therefore, the algorithm is not much efficient for practical use. It is desired to give an efficient construction of CDR for a given family of curved rays with theoretically near optimal distance bound.

There are $n!$ different CDRs in G corresponding to the ways to locate the branching pixel of each $L(k)$. Thus, it is implied that the infinite set of all diffused families of rays is mapped to $n!$ CDRs, and the inverse image of a CDR $\mathcal{T}$ is a class of families of rays within $O(\log^{1.5} n)$ distance from the set of paths of $\mathcal{T}$. It is curious to extend this observation to more general geometric objects in the plane.

References

1 C. Aistleitner, D. Bilyk, and A. Nikolov. Tusnády's problem, the transference principle, and non-uniform QMC sampling. In *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 169–180. Springer, 2016.

2 N. Bansal, D. Dadush, and S. Garg. An algorithm for Komlós conjecture matching Banaszczyk's bound. *SIAM J. Comput.*, 48(2):534–553, 2019. doi:10.1137/17M1126795.

3 N. Bansal and S. Garg. Algorithmic discrepancy beyond partial coloring. In *Proc. 49th ACM Symposium on Theory of Computing*, pages 914–926, 2017.

4 J. Beck. Balanced two-colorings of finite sets in the square I. *Combinatorica*, 1(4):327–335, 1981.

5 J. Beck and V. T. Toth. Discrepancy Theory. *Handbook of Combinatorics*, 2(Chapter 26):1406–1446, 1996.

6 G. Bohus. On the discrepancy of 3 permutations. *Random Structures & Algorithms*, 1(2):215–220, 1990.

7 M.-K. Chiu, K. Korman, M. Suderland, and T. Tokuyama. Distance bounds for high dimensional consistent digital rays and 2-d partially-consistent digital rays. In *Proc. 28th European Symposium on Algorithms*, pages 34:1–34:22, 2020.

8 I. Chowdhury and M Gibson. A characterization of consistent digital line segments in $\mathbb{Z}^2$. In *Proc. 23rd European Symposium on Algorithms*, pages 337–348, 2015.

9 I. Chowdhury and M Gibson. Constructing consistent digital line segments. In *Proc. 12th Latin American Theoretical Informatics Conference*, pages 263–274, 2016.

10 T. Christ, D. Pálvölgyi, and M. Stojaković. Consistent digital line segment. *Discrete & Computational Geometry*, 47-4:691–710, 2012.

11 J. Chun, K. Kikuchi, and T. Tokuyama. Consistent digital curved rays and pseudoline arrangements. In *Proc. 27th European Symposium on Algorithms*, pages 32:1–32:16, 2019.

12 J. Chun, M. Korman, M. Nöllenburg, and T. Tokuyama. Consistent digital rays. *Discrete & Computational Geometry*, 42-3:359–378, 2009.

13 Kunal Dutta. On shallow packings and Tusnády's problem. *CoRR*, abs/2109.05693, 2021. arXiv:2109.05693.

14 R. Klette and A. Rosenfeld. Digital straightness – a review. *Discrete Applied Math.*, 139:197–230, 2004.

15 M.G. Luby. Grid geometries which preserve properties of Euclidean geometry: A study of graphics line drawing algorithms. In *NATO Conference on Graphics/CAD*, pages 397–432, 1987.

16 J. Matoušek. *Geometric discrepancy: An illustrated guide.* Springer Science & Business Media, 1999.

17 A. Nikolov. Tighter bounds for the discrepancy of boxes and polytopes. *Mathematika*, 63(3):1091–1113, 2017.

A Appendix

A.1 Geometric Primitives

Although the existence of the CDR is given mathematically by using abstract properties of the ray family, the θ-threshold rounding algorithm needs computation of the weight w and the sequence θ. Therefore, necessary primitive geometric operations (which is called *geometric primitives*) must be executed using information of the ray family.

Given $s = (z_s, w_s)$ and $t = (z_t, w_t)$ in the (z, w)-space, we say s is *higher* than t with respect to $\mathcal{F}^*$ if there exists a gradient curve (called *separating curve*) $\varphi_C : \{(z, w(q_C(z))) \mid 0 < z \leq n\}$ such that $w(q_C(z_s)) < w_s$ and $w(q_C(z_t)) \geq w_t$. If s and t are on the same gradient curve, we say they have the same height.

The following two geometric primitives are necessary for the algorithm.

1. Given $p \in \Delta$, compute the weight $w(p)$ with a sufficient precision so that necessary comparisons in the algorithm can be done properly.
2. Given s and t in the (z, w)-plane, decide which is higher (or they have the same height) with respect to $\mathcal{F}^*$.

A given set of points in the (z, w)-plane can be sorted with respect to the height by using the second primitive. This enables to identify a range space of pseudo-rectangles to that of axis-parallel rectangles combinatorially.

We assume that each geometric primitive can be done in polynomial time in n in order to guarantee the polynomial time complexity for computing a CDR.

The computation of the weight $w(p)$ needs locally differentiable representations of rays, and the computation of $q_C(z)$ needs solution of equations as shown in the examples given below.

A.2 Examples

In the following examples, the geometric primitives need numerical computation such as solution of non-algebraic equations.

▶ **Example 20.** Consider an increasing differentiable function $f(x)$ such that $f(0) = 0$. Then, the family $\mathcal{F} = \{C_a : y = af(x) \mid a \in [0, \infty]\}$ is a smooth ray family, if we consider C_∞ as the vertical line $x = 0$. Given any $p = (x_0, y_0) \in \Delta$ for $x_0 > 0$, C_{a_0} for $a_0 = \frac{y_0}{f(x_0)}$ is the unique ray going through p, and $w(p) = \frac{1}{1+a_0 f'(x_0)} = \frac{f(x_0)}{f(x_0)+y_0 f'(x_0)}$. The family $\mathcal{F}$ is diffused if $f(x)$ is a concave function.

For example, the sigmoid function $\sigma(x) = \frac{1-e^{-x}}{1+e^{-x}}$ is a concave function for $x \geq 0$. Thus, the family $\mathcal{F} = \{C_a^{\text{sig}} : y = a\sigma(x) \mid a \in [0, \infty]\}$ is a diffused family. The derivative at $p = (x, y)$ is $y' = 2a\frac{e^{-x}}{(1+e^{-x})^2}$, which equals $\frac{2ye^{-x}}{1-e^{-2x}}$ and hence $w(p) = \frac{1-e^{-2x}}{1-e^{-2x}+2ye^{-x}}$.

The x-coordinate value of the point $q_C(z)$ for $C = C_a^{\text{sig}}$ is the root of the equation $x + a\frac{1-e^{-x}}{1+e^{-x}} = z$, and it does not have an explicit analytic expression. Thus, the geometric primitive concerning $\varphi_C(z)$ requires substantial numerical computation.

▶ **Example 21.** For $0 < a \leq 1$, define the function $F_a(x) = (1-a)n \ \sin \frac{\pi x}{2na}$ for $0 \leq x \leq na$. We define $C_a^{\text{sin}} : y = F_a(x)$ for $a > 0$, and C_0^{sin} is defined to be the y-axis. Then, $\mathcal{F} = \{C_a^{\text{sin}} \mid a \in [0, 1]\}$ is a family of (increasing segments of) sine curves in Δ. It is a diffused ray family, and we can apply our algorithm. The weight $w(p)$ for $p = (x, y)$ is not explicitly expressed by using elementary functions of x and y, and should be computed numerically.

A.3 Preliminary implementation and experiment

We give a preliminary experimental report of an implementation of the proposed θ-threshold rounding algorithm. The homogeneous polynomial ray families $\mathcal{F}_j = \{y = ax^j \mid a \in [0, \infty]\}$ for $j = 2, \ldots, 6$ are considered as the ray families in the experiment. We varied the grid size n in the range $n = 2^k$ $(1 \leq k \leq 14)$ to see the dependency of the maximum distance error on n.

As stated in the concluding remarks, it seems to be difficult to implement the SDP method with the theoretical $O(\log^2 n)$ combinatorial discrepancy bound so that it gives a practically good solution. Moreover, $n^{1/4} < \log^2 n$ if $n < 2^{44}$ (the base of logarithm is 2), and the SDP method needs $O(n^3)$ time, which is not feasible for a large n.

Hence, for the preliminary implementation, we have given a more casual method to attain an $O(n^{1/4})$ discrepancy. The hypergraph coloring is done by using the low-stabbing matching based on the k-d tree data structure on the point set described in [16]. This gives a randomized algorithm to attain an $O(n^{1/4})$ expected bound for the combinatorial discrepancy. To transfer this discrepancy bound, we apply the transference principle procedure given in [16] starting with $n^{1.5} = 2^{3k/2}$ grid points.

We measured the maximum Hausdorff distance between rays and digital rays as shown in Figure 6. The chart shows the tendency of increase of the error is about $2n^{1/4}$ to support the theory. From the chart, we can observe that the distance error is almost independent of the choice of the ray family. If $n = 14$, the grid size (width) n is 16384, and the maximum distance error is about $25 < \frac{n}{640}$ pixels, which is 0.05 inch in the 32 inch display.

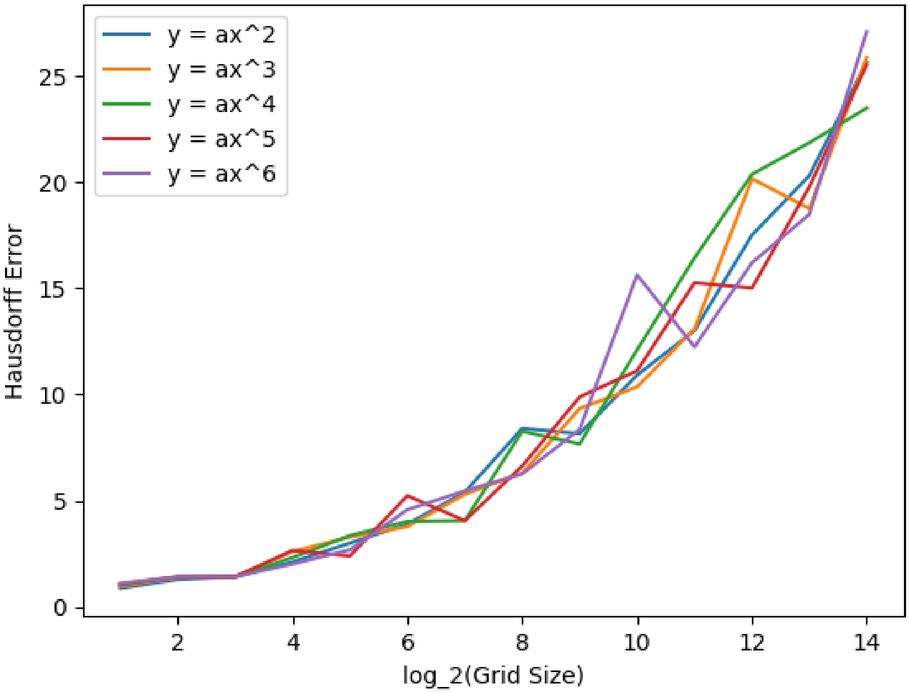


Figure 6 Distance error of the proposed algorithm.

The most expensive routine in the experiment is the measurement of the Hausdorff distance, which needs $O(n^2)$ time (in proportional to the number of pixels of the grid) if we want to measure exactly.

We compared our method with the θ-threshold rounding using the van der Corput sequence as θ, which is equivalent (although the description is different) to the previous method of Chun et al. [11]. Figure 7 shows that the distance error is about half in the van der Corput method compared with ours.

The chart shows that van der Corput method is experimentally better by a factor of approximately 2 than the low-stabbing matching method for the families considered in our experiment in the range $n \leq 2^{14}$.

This implies that although the van der Corput-Hammersley point set P gives only $O(\sqrt{n \log n})$ theoretical discrepancy bound of $D(P, \mathcal{A}_{\mathcal{F}^*})$ in the worst case, the discrepancy is practically better for most of curve families.

Therefore, although the results using the transference from the combinatorial discrepancy is theoretically better, it might be advantageous to use the van der Corput sequence (or its variants) to construct CDRs in practice.

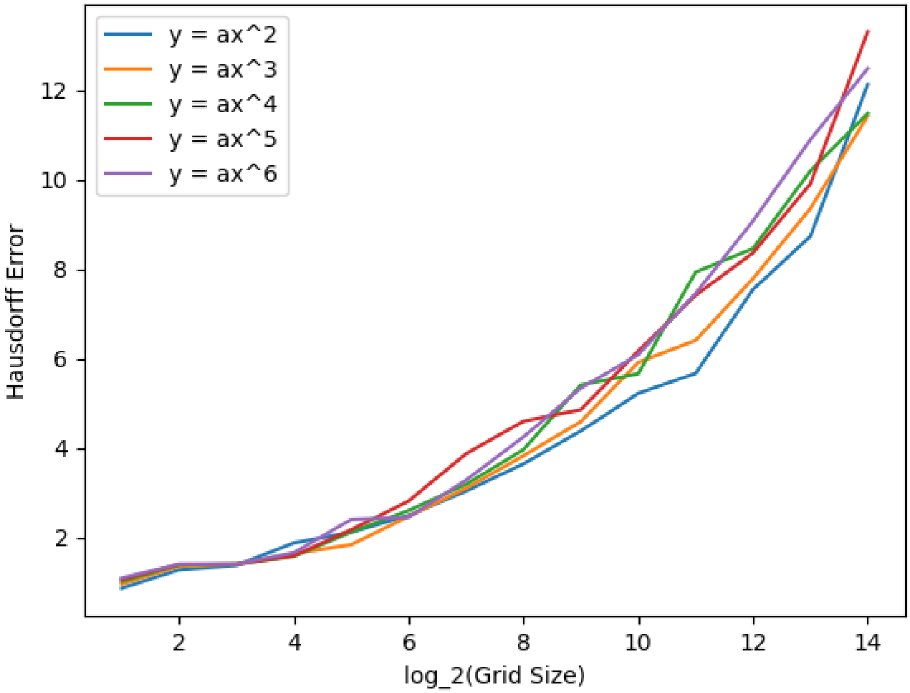


Figure 7 Distance error of the CDR using the van der Corput sequence.

Sharp Indistinguishability Bounds from Non-Uniform Approximations

Christopher Williamson
The SW7 Group, Hong Kong, China

Abstract

We study the basic problem of distinguishing between two symmetric probability distributions over n bits by observing k bits of a sample, subject to the constraint that all $(k-1)$-wise marginal distributions of the two distributions are identical to each other. Previous works of Bogdanov et al. [3] and of Huang and Viola [8] have established approximately tight results on the maximal possible statistical distance between the k-wise marginals of such distributions when k is at most a small constant fraction of n. Naor and Shamir [12] gave a tight bound for all k in the special case $k = n$ and when distinguishing with the OR function; they also derived a non-tight result for general k and n. Krause and Simon [9] gave improved upper and lower bounds for general k and n when distinguishing with the OR function, but these bounds are exponentially far apart when $k = \Omega(n)$. In this work we provide sharp upper and lower bounds on the maximal statistical distance that hold for all k and n. Upper bounds on the statistical distance have typically been obtained by providing *uniform* low-degree polynomial approximations to certain higher-degree polynomials. This is the first work to construct suitable non-uniform approximations for this purpose; the sharpness and wider applicability of our result stems from this non-uniformity.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases bounded indistinguishability, randomness, secret sharing, polynomial approximation

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.59

1 Introduction

We consider pairs of distributions μ and ν over $\{0,1\}^n$. The distributions μ and ν are said to be perfectly j-wise indistinguishable if for any subset $S \subseteq [n]$ of size at most j, the marginal distributions μ_S and ν_S over indices in S are identically distributed. The distributions are k-wise reconstructible with advantage ϵ (alternatively, ϵ-distinguishable) if there exists a set $S \subseteq [n]$ of indices of size k and a statistical test $T : \{0,1\}^{|S|} \to \{0,1\}$ such that

$$|\mathbb{E}_{X\sim\mu}[T(X|_S)] - \mathbb{E}_{Y\sim\nu}[T(Y|_S)]| \geq \epsilon,$$

where $X|_S$ is the restriction of random variable X to the bits located at the indices in S. Equivalently, μ and ν are j-wise indistinguishable if all size $\leq j$ marginal distributions have 0 total variation distance; μ and ν are k-wise reconstructible with advantage ϵ if any of the k-wise marginals have total variation distance at least ϵ. The distributions are symmetric if μ and ν are invariant under permutation (see definitions in Section 2); for such distributions the size of S is relevant for distinguishing but not the choice of indices.

Cryptographic motivation

Work of Bogdanov et al. [2] considered this notion of indistinguishability as a way to capture cryptographic secret sharing schemes in a minimal setting. Their observation was that a single bit secret can be shared by sampling n bits from μ or from ν, depending on the secret: the j-wise indistinguishability of the distributions provides a security guarantee that any size $\leq j$ coalition of colluding parties learn nothing about the secret from their joint shares. The secret reconstruction function for the scheme is a test T applied over the shares of sufficiently many (possibly all) parties. A key question in their work was how large j could be taken so that there exists a T that is both computable with AC^0 circuits and has reconstruction advantage $\epsilon = \Omega(1)$ against some pair of j-wise indistinguishable distributions. The special case of $T = \mathsf{OR}$ captures the notion of visual cryptography, introduced by Naor and Shamir [12] in 1994. Their work considered j-wise indistinguishable distributions and attempted to maximise the reconstruction advantage of OR taken over $j+1$ bits. They gave a tight bound for all j in the special case $j = n-1$; they also derived a non-tight result for general j and n. Krause and Simon [9] gave improved upper and lower bounds for general j and n, but these bounds are exponentially far apart when $j = \Omega(n)$. In this work (as in [4], [8]), we consider the statistical distance between the distributions, which includes the study of tests T that are not in AC^0 and reconstruction advantage ϵ that may be vanishing. These results can be interpreted as tight existence or non-existence conditions for 1-bit secret sharing schemes of arbitrary reconstruction complexity.

Approximate degree motivation

The work [2] largely proceeded by a connection to the theory of approximate degree of Boolean functions. The ϵ-approximate degree of a Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$, denoted $\widetilde{\deg}_\epsilon(f)$, is the least degree of a multivariate real-valued polynomial p such that $|p(x) - f(x)| \le \epsilon$ for all inputs $x \in \{0,1\}^n$. This quantity has received significant attention, owing to its polynomial equivalence to many other complexity measures including sensitivity, exact degree, deterministic and randomized query complexity [14], and quantum query complexity [5]. By linear programming duality, f has ϵ-approximate degree more than j if and only if there exists a pair of probability distributions μ and ν over $\{0,1\}^n$ such that μ and ν are j-wise indistinguishable and n-wise reconstructible with advantage 2ϵ by f (see for example [2, 17, 6]). The approximate degree of all symmetric Boolean functions was resolved in the constant-error regime $\epsilon = \Theta(1)$ by Paturi [15] and in the general error regime by de Wolf [7] using an argument based on quantum algorithms (see also Sherstov [18] and Bun and Thaler [6]). This implies tight upper and lower bounds on the ability of any given symmetric Boolean function to reconstruct from indistinguishable distributions when given access to a full sample of n bits. In this work, we consider reconstruction when given access to a subset of the bits.

Prior work

Works of Bogdanov et al. [3] and of Huang and Viola [8] extended the study of the indistinguishability of symmetric distributions to the setting of reconstructing with a subset of indices. They considered the extent to which symmetric j-wise indistinguishable distributions must have statistically close k-wise marginals for $k > j$. In particular, [3] shows that if μ and ν are symmetric over n-bit strings and perfectly j-wise indistinguishable, then the statistical distance between k-wise marginals is at most $O(j^{3/2}) \cdot e^{-j^2/1156k}$ for all $j < k \le n/64$. The analogous result in [8] gives a similar bound and also applies to k at most some (unspecified) constant fraction of n. A matching lower bound given in [3] shows that there exists a pair of distributions that are j-wise indistinguishable but reconstructable with the OR_k function with advantage at least $k^{-1/2} \cdot e^{-O(-j^2/k)}$. This lower bound extends to all $j < k \le n$.

We note that in the sharp reconstruction setting where $k = j+1$, the upper bound $O(j^{3/2}) \cdot e^{-j^2/1156k}$ solely demonstrates a behaviour of monotonic exponential decrease in k. Extension of this upper bound to $k > n/64$ is of particular interest because it would illuminate and quantify the behaviour that the maximum statistical distance must reach a minimum at some $k \in [n/64, n]$ and start to increase once k becomes large enough. Specifically, we know this behaviour changes because the XOR_n function can distinguish perfectly between a pair of symmetric distributions that are $(n-1)$-wise indistinguishable (the two distributions are uniform over n bits, conditioned on the sum of all n bits being either even or odd).

In addition, the lower bound for OR in [3] does apply to k up to n but is unlikely to come close to matching any upper bound on statistical distance since OR is a weak statistical distinguisher and the bound is also monotonically decreasing.

Our contribution

In the present work, we extend the results of [3] and [8] to the setting of parameters where k may range freely from 0 to $n-1$ (the case $k = n$ is trivial in light of the XOR example above). We consider the sharp threshold reconstruction setting $j = k-1$ (see prior work section) and our results are tight up to polynomial factors.

▶ **Theorem 1.** *There exists an absolute constant c such that for any $k \in [0, n-1]$ and any pair of symmetric $(k-1)$-wise indistinguishable distributions μ, ν over $\{0,1\}^n$, the statistical distance between all k-wise marginals of μ and ν is at most:*

$$O(n^c) \cdot \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}$$

▶ **Theorem 2.** *For any $k \in [0, n-1]$, there exists a statistical test $T : \{0,1\}^k \to \{0,1\}$ and a pair of symmetric $(k-1)$-wise indistinguishable distributions μ, ν such that the reconstruction advantage of T is at least*

$$\frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}.$$

An indistinguishability game. The upper and lower bounds of Theorems 1 and 2 allow us to approximately find the value of k so that the task of using k bits to distinguish symmetric perfectly $(k-1)$-wise indistinguishable distributions over $\{0,1\}^n$ is most difficult. For n an integer fixed in advance, this can be expressed in terms of a game:

- Player 1 selects integer $k < n$.
- Player 2 chooses a pair μ, ν of symmetric, perfectly $(k-1)$-wise indistinguishable distributions over $\{0,1\}^n$.
- The payoff p_2 of Player 2 is defined as the statistical distance between the k-wise marginals of μ and ν and the payoff of Player 1 is $p_1 = 1 - p_2$.

We show that the optimal strategy of Player 1 is essentially to select $k = 0.6n$. More precisely:

▶ **Corollary 3.** *In the indistinguishability game defined above, let k^* be an optimal strategy of Player 1. Then, for any arbitrarily small positive constant ϵ, and n sufficiently large, k^* satisfies:*

$$|k^* - 0.6n| \leq \epsilon n.$$

A naive assumption would be that for a fixed n and k ranging in $[0, n]$, the behaviour of the quantity being bound in Theorems 1 and 2 is symmetric around the centre of the interval $[0, n]$, or namely that the optimal strategy of Player 1 is to choose $k = n/2$. Corollary 3 demonstrates that this is false.

Finally, Theorem 1 can be translated into Fourier analytic language. We consider a function $f : \{-1,1\}^n \to \mathbb{R}$ and write it in the Fourier basis as $f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \cdot \prod_{i \in S} x_i$, where $\hat{f}(S) = \mathbb{E}_{x \in \{-1,1\}^n}[f(x) \cdot \prod_{i \in S} x_i]$.

▶ **Corollary 4.** *Let $f : \{-1,1\}^n \to \mathbb{R}$ be a real-value Boolean function that is symmetric ($|S| = |S'| \implies \hat{f}(S) = \hat{f}(S')$), has no low-degree terms ($|S| \le k-1 \implies \hat{f}(S) = 0$), and has low 1-norm ($\sum_{z \in \{-1,1\}^n} |f(z)| = 1$). Then, for any S of size k, we have:*

$$\hat{f}(S) \le O(n^c) \cdot \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}.$$

Techniques and roadmap

The established technique to develop indistinguishability upper bounds for symmetric distributions is to decompose an arbitrary statistical test into a small basis using the fact that without loss of generality, the best test is a symmetric function. The basis we work over is Q_w for $w = 0, ..., k$ where Q_w is a Boolean function that observes k bits and accepts if and only if the observed Hamming weight is exactly w. Providing a low-degree polynomial approximation to Q_w rules out the existence of distributions that can be reconstructed with Q_w. In practice, a Minsky-Papert symmetrization (see Fact 13 for details) is applied to Boolean function Q_w to reduce the problem of its approximation to a problem of approximating a real-valued univariate polynomial with a lower degree polynomial. Due to the discrete domain of Q_w (the Boolean cube), the univariate approximations need not be uniform, but are instead over a set of separated points on the real line, each representing a Hamming weight of input (for example, $-1, -1 + 2/n, ..., 1 - 2/n, 1$). However, this fact is not typically exploited, and previous works have constructed approximations that have low error over the entire interval $[-1, 1]$.

Prior works have not obtained statistical distance upper bounds for k close to n because the approach taken to approximating (the symmetrized version of) Q_w has been to use Chebyshev polynomials to provide uniform approximations to Q_w over all of $[-1, 1]$ instead of discrete approximations. This strategy breaks down for large k because the difficulty of uniform approximations diverges from the difficulty of the approximation over the relevant discrete set. This observation motivates the use of discrete Chebyshev polynomials (also known as Gram polynomials) to construct approximations that yield upper bounds on the maximum statistical distance.

We provide a lower bound on statistical distance based on hardness of approximation with discrete Chebyshev polynomials. This follows from their orthogonality and from linear programming duality. We believe that prior techniques via orthogonality of (non-discrete) Chebyshev polynomials could be used to show this result (indeed the lower bound result from [3] applies to all k up to n and the technique they use should be extendable to distinguishers more powerful than OR).

2 Preliminaries

We use the standard notion of statistical distance (total variation distance) throughout this paper. Specifically, the statistical distance between μ and ν is $\frac{1}{2}||\mu - \nu||_1$, or $\frac{1}{2}\sum_{z \in \{0,1\}^n} |\Pr_{X \sim \mu}[X = z] - \Pr_{Y \sim \nu}[Y = z]|$. This is equivalent to the reconstruction advantage of the optimal statistical test for μ and ν that observes all n bits.

We will be working with polynomial approximations over different discrete sets of points. We define D_n^{out} as the set of points $\{-1, -1{+}2/n, ..., 1\}$ and D_n^{in} as $\{-1{+}1/n, -1{+}3/n, ..., 1{-}1/n\}$. It is easy to check that $|D_n^{\text{out}}| = n{+}1$, that $|D_n^{\text{in}}| = n$. Further, we have the basic relationships:

$$D_n^{\text{out}} = \left\{\frac{n+1}{n} \cdot x : x \in D_{n+1}^{\text{in}}\right\} \tag{1}$$

$$D_n^{\text{in}} \subset \left\{x - \frac{1}{n} : x \in D_n^{\text{out}}\right\} \tag{2}$$

For simplicity we will use $\lesssim$ to hide factors polynomial in n.

Symmetric distributions and functions

Let $f : \{0,1\}^n \to \mathbb{R}$ be a function. We say that f is symmetric if the output of f depends only on the Hamming weight of its input. A probability distribution μ is symmetric if the corresponding probability mass function mapping inputs to probabilities is a symmetric function. We also will need two further facts about distinguishing symmetric distributions. Proofs of these appear in [3].

▶ **Fact 5.** *Suppose that μ is a symmetric distribution over $\{0,1\}^n$. For $S \subseteq \{0, ..., n\}$, let $\mu|_S$ denote the restriction of μ to the indices in S. Then, $\mu|_S$ is also symmetric.*

▶ **Fact 6.** *Suppose that μ and ν are symmetric distributions over $\{0,1\}^n$. Then without loss of generality, the best statistical test $Q : \{0,1\}^n \to [0,1]$ for distinguishing between μ and ν is a symmetric function. In particular, we have:*

$$\max_{\textit{symmetric } Q}\{\mathbb{E}_{X\sim\mu}[Q(X)] - \mathbb{E}_{Y\sim\nu}[Q(Y)]\} = \max_{Q}\{\mathbb{E}_{X\sim\mu}[Q(X)] - \mathbb{E}_{Y\sim\nu}[Q(Y)]\}.$$

2.1 Discrete Chebyshev polynomials

The discrete Chebyshev polynomials, for parameter n are a family of real polynomials $\{\phi_d\}_{d=0,\dots,n-1}$. Borrowing notation from [1], we have that the polynomials have the following properties:

- The family of polynomials $\{\phi_d\}_{d=0,\dots,n-1}$ are orthogonal with respect to the bilinear form given by

$$(\phi_i, \phi_j) := \frac{1}{n} \cdot \sum_{x \in D_n^{\text{in}}} \phi_i(x) \cdot \phi_j(x) \tag{3}$$

- For each d:

$$||\phi_d|| := (\phi_d, \phi_d)^{1/2} = 1 \tag{4}$$

- For each d:

$$\deg(\phi_d) = d \tag{5}$$

- The polynomials satisfy the recurrence:

$$\phi_d(x) = 2\alpha_{d-1} \cdot x \cdot \phi_{d-1}(x) - \frac{\alpha_{d-1}}{\alpha_{d-2}} \cdot \phi_{d-2}(x) \tag{6}$$

$$\alpha_{d-1} = \frac{n}{d} \cdot \left(\frac{d^2 - 1/4}{n^2 - d^2} \right)^{1/2} \tag{7}$$

where we have $\phi_0 = 1$, $\phi_{-1} = 0$, and $\alpha_{-1} = 1$.

Every degree $k < n$ polynomial $p\colon \mathbb{R} \to \mathbb{R}$ has a unique expansion in the discrete Chebyshev basis:

$$p(t) = \sum_{d=0}^{k} c_d \phi_d(t),$$

where $c_0, \ldots, c_k$ are the *discrete Chebyshev coefficients* of p.

2.2 Bounds on factorials and binomial coefficients

We will make use of double factorials, which are given by:

$$n!! := \prod_{i=0}^{\lfloor n/2 \rfloor} n - 2i \tag{8}$$

and satisfy, when n is even:

$$n!! = 2^{n/2} \cdot (n/2)! \tag{9}$$

For n odd, we simply observe that $(n-1)!! \lesssim n!! \lesssim (n+1)!!$ and apply the bound in (9).

We will bound factorials using

$$n! = \Theta(\sqrt{n}) \cdot \left(\frac{n}{e} \right)^n \tag{10}$$

and the central binomial coefficient using

$$\binom{2n}{n} = \Theta(1/\sqrt{n}) \cdot 2^{2n} \tag{11}$$

2.3 Approximate degree of Boolean functions

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. We will use $\widetilde{\deg}_\epsilon(f)$ to denote the minimum degree of any real polynomial $p : \{0,1\}^n \to \mathbb{R}$ that approximates f to within ϵ at every point in $\{0,1\}^n$.

Paper organisation. In Section 3, we prove a lemma about the expression of monomials in the discrete Chebyshev basis. In Section 4 we construct discrete approximations to the monomial and prove a complementary hardness of approximation result in Section 5. In Section 6 we quantify the precise approximation problem that we need to solve and justify this using symmetrization and linear programming duality techniques. Sections 7 and 8 justify Theorem 1 and Theorem 2, respectively. In the Appendix, we handle proofs of some technical claims and proofs of Corollaries 3 and 4.

3 Monomials in the discrete Chebyshev basis

▶ **Lemma 7.** *Fix integer $k < n$. Let C be the leading coefficient in the expansion of x^k in the discrete Chebyshev polynomial basis with parameter n. Then, C satisfies:*

$$\frac{1}{(2n)^k} \cdot \sqrt{\frac{(n+k)!}{(n-k)!}} \lesssim C \lesssim \frac{1}{(2n)^k} \cdot \sqrt{\frac{(n+k)!}{(n-k)!}}$$

Proof. From the recurrence definition of the discrete Chebyshev polynomials in Equation 6, we have that

$$x \cdot \phi_i = \frac{1}{2\alpha_{i-1}} \phi_{i+1} + \frac{1}{2\alpha_{i-2}} \phi_{i-1}$$

Application of this recursion, along with the base case $\phi_0 = 1$, yields that the discrete Chebyshev representation of x^k has its highest degree coefficient given by

$$C = \prod_{i=0}^{k-1} \frac{1}{2\alpha_i} = 2^{-k} \cdot \prod_{i=0}^{k-1} \frac{1}{\alpha_i} \tag{12}$$

From the definition of the α_i in Equation 7, we have that

$$\begin{aligned}\prod_{i=0}^{k-1} \alpha_i &= \frac{n^{k-1}}{(k-1)!} \sqrt{\frac{(1^2-1/4)(2^2-1/4) \cdot \ldots \cdot ((k-1)^2-1/4)}{(n^2-1^2)(n^2-2^2) \cdot \ldots \cdot (n^2-(k-1)^2)}} \\ &= \frac{n^{k-1}}{(k-1)!} \sqrt{\frac{(1^2-1/4)(2^2-1/4) \cdot \ldots \cdot ((k-1)^2-1/4)}{(n+1)(n-1)(n+2)(n-2) \cdot \ldots \cdot (n+k-1)(n-k+1)}} \\ &= n^{k-1} \cdot \sqrt{\frac{n!(n-k)!}{(n-1)!(n+k-1)!}} \cdot \sqrt{\frac{(1^2-1/4)(2^2-1/4) \cdot \ldots \cdot ((k-1)^2-1/4)}{(k-1)^2 \cdot \ldots \cdot 1^2}} \\ &= n^{k-1} \cdot \sqrt{\frac{n \cdot (n-k)!}{(n+k-1)!}} \cdot \sqrt{\prod_{i=1}^{k-1} \frac{(i-1/2)(i+1/2)}{i^2}}.\end{aligned}$$

Application of the upper bound in Claim 20 yields that $\prod_{i=0}^{k-1} \alpha_i \le n^{k-1} \cdot \sqrt{\frac{n \cdot (n-k)!}{(n+k-1)!}}$. This, in conjunction with Equation 12, justifies the lower bound in the statement of this lemma. For the upper bound, application of the lower bound in Claim 20 yields that $\prod_{i=0}^{k-1} \alpha_i \ge \frac{3}{4(k-1)^2} \cdot n^{k-1} \cdot \sqrt{\frac{n \cdot (n-k)!}{(n+k-1)!}}$, which completes the proof in light of Equation 12. ◀

4 Discrete approximations for the monomial

▶ **Corollary 8.** *Fix integers $k < n$. There exists a degree at most $k-1$ polynomial approximation for the monomial x^k over D_n^{out} with error ϵ satisfying:*

$$\epsilon \lesssim (2(n+1))^{-k} \cdot \sqrt{\frac{(n+k+1)!}{(n-k+1)!}}$$

Proof. It suffices to provide a degree at most $k-1$ approximation p to the monomial $(\frac{n+1}{n} \cdot x)^k$ over D_{n+1}^{in} because then the degree at most $k-1$ approximation $p' := p(\frac{n}{n+1} \cdot x)$ will be an approximation to x^k over D_n^{out}, by Equation 1. By Lemma 7, the expansion of $(\frac{n+1}{n} \cdot x)^k$ in the Gram basis with parameter $n+1$ is given by

$$C_k \cdot \phi_k + C_{k-1} \cdot \phi_{k-1} + \ldots + C_0 \cdot \phi_0,$$

where $C_k \lesssim \left(\frac{n+1}{n}\right)^k \cdot \frac{1}{(2(n+1))^k} \cdot \sqrt{\frac{(n+k+1)!}{(n-k+1)!}} \lesssim \frac{1}{(2(n+1))^k} \cdot \sqrt{\frac{(n+k+1)!}{(n-k+1)!}}$. By Equation 4 and Cauchy-Schwarz, $\max_{D^{\text{in}}_{n+1}} \phi_k \lesssim 1$ and the corollary follows by taking the approximation $\sum_{i=0}^{k-1} C_i \cdot \phi_i$. ◀

Comparison to the uniform approach

Newman and Rivlin [13], and Sachdeva and Vishnoi [16] showed that any degree $\leq k-1$ uniform approximation to the monomial x^k over $[-1,1]$ will have error 2^{-k+1} (and that this is tight). For large enough values of n and k, the upper bound in the statement of Corollary 8 is smaller. In Section 7 we see that the bound 2^{-k+1} is insufficient to get a non-trivial indistinguishability upper bound for k close to n.

5 Hardness of discrete monomial approximations

▶ **Corollary 9.** *Fix integers $k < n$. Any degree at most $k-1$ polynomial approximation to the monomial x^k must have error (pointwise over D_n^{out}) at least:*

$$(2n)^{-k} \cdot \sqrt{\frac{(n+k)!}{(n-k)!}}$$

The proof of the main corollary of this section appears at the end of this section after two lemmas have been established.

▶ **Lemma 10.** *Let p be a degree k polynomial with degree k coefficient C in the discrete Chebyshev basis with parameter n. Then, any degree $k-1$ approximating polynomial will have error at least $|C|$ at some point over D_n^{in}.*

Proof. Let q be any degree at most $k-1$ polynomial. Let c_i be the degree i coefficient in the discrete Chebyshev representation of $p-q$, and note that because the degree of q is at most $k-1$, we have that $c_k = C$. By orthogonality of the discrete Chebyshev polynomials, and Equations 3 and 4,

$$\mathbb{E}_{t\sim D_n^{\text{in}}}[(p(t)-q(t))^2] = c_0^2 + \sum_{d=1}^{k}(c_d)^2\,\mathbb{E}_{t\sim D_n^{\text{in}}}[\phi_d(t)^2] \geq c_k^2 = C^2.$$

It follows that the approximation error $|p(t)-q(t)|$ must exceed $|C|$ for some $t \in D_n^{\text{in}}$. ◀

▶ **Lemma 11.** *Let p be a degree k polynomial such that for any degree at most $k-1$ polynomial q, $\max_{t\in D_n^{in}}\{|p(t)-q(t)|\} \geq \epsilon$. Then, for any degree $k-1$ polynomial q', $\max_{t\in D_n^{out}}\{|p(t)-q'(t)|\} \geq \epsilon$.*

Proof. We consider the contrapositive and show that existence of a degree at most $k-1$ polynomial $\tilde{p}$ for p over D_n^{out} with error at most ϵ implies an approximation for p over D_n^{in} with the same degree and error parameters. We have that $\tilde{p}(x+1/n)$ is a degree $k-1$ ϵ−approximation of $p(1+1/n)$ over $\{x - \frac{1}{n} : x \in D_n^{\text{out}}\} \supset D_n^{\text{in}}$, where the set relation follows from Equation 2. Our approximation is then $\tilde{p}(t+1/n) + p(t) - p(t+1/n)$, which is degree $k-1$ because $p(t) - p(t+1/n)$ is degree $k-1$. We have that $\max_{t\in D_n^{\text{in}}} |\,(\tilde{p}(t+1/n)+p(t)-p(t+1/n)) - p(t)| = \max_{t\in D_n^{\text{in}}} |\tilde{p}(t+1/n) - p(t+1/n)| \leq \epsilon$. ◀

Proof of Corollary 9. Lemma 7 and Lemma 10 imply that any degree at most $k-1$ polynomial approximation to x^k must have error at least $(2n)^{-k} \cdot \sqrt{\frac{(n+k)!}{(n-k)!}}$ over D_n^{in}. The corollary then follows from Lemma 11. ◀

6 Symmetrization and duality

Our main upper and lower bounds will be justified by reducing to an approximation theoretic question using a linear programming duality relation.

▷ Claim 12 (see, for example, Theorem 1.2 in [2]). $\widetilde{\deg}_{\epsilon/2}(F) \geq k$ if and only if there exists a pair of perfectly k-wise indistinguishable distributions μ, ν over $\{0,1\}^n$ such that $\mathbb{E}_{X\sim\mu}[F(X)] - \mathbb{E}_{Y\sim\nu}[F(Y)] \geq \epsilon$.

We are interested in Boolean functions as statistical tests that witness k bits of a sample from a distribution. To this end, let Q_w denote the function on $\{0,1\}^k$ that outputs 1 if and only if the Hamming weight is exactly w. We also use $Q_w(x|_S)$ to represent Q_w when evaluated on a string of n bits, where $x|_S$ is the restriction of the n bits to the k indices in the set $S \subseteq [n]$.

▶ **Fact 13.** *Let $S \subseteq [n]$ be any set of size k. There exists a univariate polynomial p_w of degree at most k such that the following holds. For all $t \in D_n^{out}$, $p_w(t) = \mathbb{E}_Z[Q_w(Z|_S)]$ where Z is a uniformly random string of n bits conditioned on having Hamming weight $\phi^{-1}(t) = (1-t)n/2 \in \{0,1,\ldots,n\}$.*

Proof. This statement is a simple extension of Minsky and Papert's classic symmetrization technique [11] and also appears in [3]; we reproduce the proof here for convenience. Minsky and Papert showed that for any polynomial $P\colon \{0,1\}^n \to \mathbb{R}$, there exists a univariate polynomial p of degree at most the total degree of P, such that for all $i \in \{0,\ldots,n\}$, $p(i) = \mathbb{E}_{|x|=i}[P(x)]$. Apply this result to $P(x) = Q_w(x|_S)$ and let $p_w(t) = p(\phi^{-1}(t)) = p\left((1-t)n/2\right)$. The fact then follows from the observation that the total degree of $Q_w(x|_S)$ is at most k, since this function is a k-junta. ◀

▶ **Corollary 14.** *Suppose that for all degree $\leq k-1$ polynomials q we have that $\max_{t\in D_n^{out}}\{|p_w(t) - q|\} \geq \epsilon$. Then, $\widetilde{\deg}_\epsilon(Q_w(x|_S)) \geq k-1$.*

Proof. We prove the contrapositive. Let $\tilde{Q}$ be a degree at most $k-1$ approximation to $Q_w(x|_S)$ over $\{0,1\}^n$ with pointwise error strictly less than ϵ. Taking the Minsky-Papert symmetrization of $\tilde{Q}$ in conjunction with a scale and shift, yields a univariate polynomial q of degree at most $k-1$ such that $\max_{t\in D_n^{\text{out}}}\{|p_w(t) - q|\} < \epsilon$. ◀

6.1 Properties of p_w

The value $p_w(t)$ is a probability for every $t \in D_n^{\text{out}}$. Moreover, this probability must equal zero when the Hamming weight of Z is less than w or greater than $n-k+w$. Therefore p_w has k distinct zeros at the points $Z_w = Z_- \cup Z_+$, where

$$Z_- = \{-1 + 2h/n : h = 0, ..., k-w-1\}, \quad Z_+ = \{1 - 2h/n : h = 0, ..., w-1\}. \tag{13}$$

and so p_w must have the form

$$p_w(t) = C_w \cdot \prod_{z\in Z_w} (t - z) \tag{14}$$

for some C_w that does not depend on t.

▷ Claim 15. The coefficient C_w on the highest degree term of p_w in the monomial basis has absolute value:

$$\frac{\binom{k}{w}\binom{n-k}{\frac{1}{2}(n-k)}}{\binom{n}{\frac{1}{2}(n-k+2w)}} \cdot \frac{n^k \cdot (n-k)!!^2}{(n-k+2w)!! \cdot (n-2w+k)!!}$$

Proof. The polynomial p_w is of degree k with all of its zeroes lying in Z_w. We evaluate p_w at a point t' which is necessarily outside of Z_w and thus not a zero of p_w. We set:

$$t' := \frac{1}{2}\left(\max\{Z_-\} + \min\{Z_+\}\right) = \frac{k-2w}{n}$$

To evaluate $p_w(t')$, we use that the value $p_w(t')$ is the probability that $Q_w(x|_S)$ accepts given that x is chosen uniformly at random, conditioned on the event that the Hamming weight of x is exactly $\phi^{-1}(t') = \frac{1}{2}(n-k+2w)$.

$$\Pr\left[Q_w(x|_S) = 1 : |x| = \frac{1}{2}(n-k+2w)\right] = p_w(t') = C_w \cdot \prod_{z\in Z_w}(t'-z),$$

from which it follows that

$$C_w = \frac{\binom{k}{w}\binom{n-k}{\frac{1}{2}(n-k)}}{\binom{n}{\frac{1}{2}(n-k+2w)} \cdot \prod_{z\in Z_w}(t'-z)}$$

We have that:

$$\begin{aligned}\prod_{z\in Z_w}(t'-z) &= \prod_{z\in Z_-}(t'-z)\prod_{z\in Z_+}(t'-z)\\ &= (-1)^{|Z_+|}\prod_{z\in Z_+}(z-t')^2 \prod_{z\in Z_-:-z\notin Z_+}(t'-z),\end{aligned}$$

where the final equality assumes that $w \leq k/2$. This is without loss of generality; when $w > k/2$, the same calculation holds with the roles of Z_+ and Z_- reversed. From this we compute that:

$$\begin{aligned}\frac{1}{|\prod_{z\in Z_w}(t'-z)|} &= \frac{1}{(1-\frac{k-2}{n})^2\cdot(1-\frac{k-2}{n}+\frac{2}{n})^2\cdot\ldots\cdot(1-\frac{k-2}{n}+\frac{2(w-1)}{n})^2\cdot\prod_{z\in Z_-:-z\notin Z_+}(t'-z)}\\ &= \frac{n^{2w}\cdot(n-k)!!^2}{(n-k+2w)!!^2\cdot\prod_{z\in Z_-:-z\notin Z_+}(t'-z)}\\ &= \frac{n^{2w}\cdot(n-k)!!^2}{(n-k+2w)!!^2\cdot\prod_{i=0}^{k-2w-1}(\frac{k-2w}{n}+1-\frac{2i}{n})}\\ &= \frac{n^{k}\cdot(n-k)!!^2}{(n-k+2w)!!\cdot(n+k-2w)!!},\end{aligned}$$

from which the claim follows. ◁

We also will need the following fact, which is justified in the Appendix.

▶ **Lemma 16.** *The value C_w is maximized when $w = k/2$; in particular with*

$$C_{k/2} = \frac{\binom{k}{k/2}\binom{n-k}{(n-k)/2}}{\binom{n}{n/2}} \cdot \frac{n^k\cdot(n-k)!!^2}{n!!^2}$$

7 Upper bound

We begin by providing an upper bound on the distinguishing advantage of a given Q_w test.

▶ **Lemma 17.** *For any $w = 0, ..., k$ and pair of $(k-1)$-wise indistinguishable distributions, the function Q_w reconstructs with advantage ϵ, satisfying:*

$$\epsilon \lesssim \frac{(n-k)^{\frac{n-k}{2}}\cdot(n+k)^{\frac{n+k}{2}}}{2^k\cdot n^n}$$

Proof. By Corollary 8, there exists a degree $k-1$ polynomial approximation to p_w over D_n^{out} with error

$$\lesssim C_w \cdot \frac{1}{(2(n+1))^k} \cdot \sqrt{\frac{(n+k+1)!}{(n-k+1)!}} \tag{15}$$

By Claim 15 and Lemma 16 this is upper bounded by (up to poly(n) factors):

$$\frac{\binom{k}{k/2}\binom{n-k}{(n-k)/2}}{\binom{n}{n/2}} \cdot \frac{n^k \cdot (n-k)!!^2}{n!!^2} \cdot \frac{1}{(2(n+1))^k} \cdot \sqrt{\frac{(n+k+1)!}{(n-k+1)!}}$$

$$\begin{aligned}
&\lesssim \frac{k! \cdot (n-k)! \cdot n^k}{(k/2)!^2 \cdot n! \cdot 2^k} \cdot \frac{1}{(2n+2)^k} \cdot \sqrt{\frac{(n+k+1)!}{(n-k+1)!}} \\
&\lesssim \frac{(n-k)^{n-k} \cdot (2e)^k}{n^n} \cdot \frac{n^k}{2^k} \cdot \frac{1}{(2n+2)^k} \cdot \frac{(n+k+1)^{(n+k+1)/2)}}{e^k \cdot (n-k+1)^{(n-k+1)/2}} \\
&\lesssim \frac{(n-k)^{(n-k-1)/2} \cdot (n+k+1)^{(n+k+1)/2}}{2^k \cdot n^n} \\
&\lesssim \frac{(n-k)^{(n-k)/2} \cdot (n+k)^{(n+k)/2}}{2^k \cdot n^n},
\end{aligned}$$

where we have used Equations 11, 10, and 9 to bound the central binomial coefficient, the factorial, and the double factorial, respectively. ◀

Comparison to the uniform approach

We saw in Section 4 that any *uniform* approximation to the monomial would have error 2^{-k+1}. Substituting that bound into Equation 15 and carrying out the same calculation would yield an upper bound of $\frac{e^k \cdot (n-k)^{n-k}}{2^k \cdot n^{n-k}}$, which for $k \approx n$ is $\gtrsim (e/2)^n$. Because any distinguishing advantage must be at most 1, this is a vacuous bound.

7.1 Proof of upper bound: Theorem 1

▶ **Theorem 18.** *For any pair of $(k-1)$-wise indistinguishable distributions μ, ν over $\{0,1\}^n$, the statistical distance ϵ between $\mu|_k$ and $\nu|_k$ satisfies:*

$$\epsilon \lesssim \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}$$

Proof. Let T be a general distinguisher on k inputs. By Facts 5 and 6, T can be assumed to be a symmetric Boolean-valued function and has the representation $T = \sum_{w=0}^k b_w \cdot Q_w$ where each of the b_w is either 0 or 1. We bound the distinguishing advantage as follows. Recalling that μ and ν are $k-1$-indistinguishable symmetric distributions over $\{0,1\}^n$, for any set $S \subseteq [n]$ of size k we have:

$$\begin{aligned}
\mathbb{E}_{X\sim\mu}[T(X|_S)] - \mathbb{E}_{Y\sim\nu}[T(Y|_S)] &= \sum_{w=0}^k b_w \big(\mathbb{E}[Q_w(X|_S)] - \mathbb{E}[Q_w(Y|_S)]\big) \\
&\le \sum_{w=0}^k \big|\mathbb{E}[Q_w(X|_S)] - \mathbb{E}[Q_w(Y|_S)]\big| \\
&\le (k+1) \cdot \max_{w=0,\dots,k} \big|\mathbb{E}[p_w(\phi(|X|)] - \mathbb{E}[p_w(\phi(|Y|))]\big| \\
&\lesssim \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n},
\end{aligned}$$

where the final upper bound is from Lemma 17. ◀

8 Proof of lower bound: Theorem 2

▶ **Theorem 19.** *Let $Q_{k/2}$ be the statistical test over k bits that accepts if and only if the observed Hamming weight is $k/2$. There exists a pair of $(k-1)$-wise indistinguishable distributions X, Y such that the reconstruction advantage of $Q_{k/2}$ is at least ϵ, satisfying:*

$$\epsilon \gtrsim \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}.$$

Proof. By Claim 12 and Corollary 14, it suffices to show that any degree $k-1$ polynomial approximation to $p_{k/2}$ over D_n^{out} must have error at least ϵ. Lemma 11 reduces the problem further to proving hardness of approximation of $p_{k/2}$ over D_n^{in}. From Claim 15 and Lemma 7, the coefficient on the degree k term of the discrete Chebyshev representation of $p_{k/2}$ is

$$\gtrsim \frac{\binom{k}{k/2}\binom{n-k}{(n-k)/2}}{\binom{n}{n/2}} \cdot \frac{n^k \cdot (n-k)!!^2}{n!!^2} \cdot \frac{1}{(2n)^k} \cdot \sqrt{\frac{(n+k)!}{(n-k)!}},$$

which is $\gtrsim \epsilon$, by trivially applying the bounds for the central binomial coefficient, factorial, and double factorial in Section 2. The theorem then follows by applying Lemma 10. ◀

9 Future research

It would be worthwhile to explicitly construct the distributions of Theorem 2 (as done in [10] to match the non-constructive bounds in [9]). We also ask whether similar methods can be extended to work in the full setting of [3, 8] in which there may be a gap between the indistinguishability and reconstruction parameters or in the context of distributions μ, ν over Σ^n, where Σ is an alphabet of size larger than 2. Finally, we observe the interplay between results stemming from classical approximation theory (often featuring Chebyshev polynomials) and quantum algorithms (notably in the pair of papers [18, 7]) and ask whether our results can be recovered with a quantum argument.

References

1 RW Barnard, Germund Dahlquist, K Pearce, Lothar Reichel, and KC Richards. Gram polynomials and the kummer function. *Journal of approximation theory*, 94(1):128–143, 1998.

2 Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In *CRYPTO*, 2016.

3 Andrej Bogdanov, Nikhil S Mande, Justin Thaler, and Christopher Williamson. Approximate degree, secret sharing, and concentration phenomena. *RANDOM*, 2019.

4 Andrej Bogdanov and Christopher Williamson. Approximate bounded indistinguishability. In *ICALP*, 2017.

5 H. Buhrman, R. Cleve, R. de Wolf, and C. Zalka. Bounds for small-error and zero-error quantum algorithms. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, 1999.

6 Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and markov–bernstein inequalities. *Information and Computation*, 243:2–25, 2015.

7 Ronald de Wolf. A note on quantum algorithms and the minimal degree of epsilon-error polynomials for symmetric functions. *arXiv preprint arXiv:0802.1816*, 2008.

8 Xuangui Huang and Emanuele Viola. Approximate degree-weight and indistinguishability. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 26, page 85, 2019.

9 Matthias Krause and Hans Ulrich Simon. Determining the optimal contrast for secret sharing schemes in visual cryptography. In Gaston H. Gonnet and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics*, volume 1776 of *Lecture Notes in Computer Science*, pages 280–291. Springer Berlin Heidelberg, 2000.

10 Christian Kuhlmann and Hans Ulrich Simon. Construction of visual secret sharing schemes with almost optimal contrast. In *Symposium on Discrete Algorithms: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, volume 9 (11), pages 263–272, 2000.

11 Marvin Minsky and Seymour Papert. *Perceptrons.* MIT Press, Cambridge, MA, 1969.

12 Moni Naor and Adi Shamir. Visual cryptography. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 1994.

13 DJ Newman and TJ Rivlin. Approximation of monomials by lower degree polynomials. *aequationes mathematicae*, 14(3):451–455, 1976.

14 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

15 Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *ACM Symp. on the Theory of Computing (STOC)*, pages 468–474, 1992.

16 Sushant Sachdeva and Nisheeth Vishnoi. Approximation theory and the design of fast algorithms. *arXiv preprint*, 2013. `arXiv:1309.4882`.

17 Alexander A. Sherstov. The pattern matrix method for lower bounds on quantum communication. In *40th ACM Symp. on the Theory of Computing (STOC)*, pages 85–94, 2008.

18 Alexander A Sherstov. Approximate inclusion-exclusion for arbitrary symmetric functions. *Computational Complexity*, 18(2):219–247, 2009.

A A technical claim

▷ Claim 20. Let $v = \prod_{i=1}^{k} \frac{(i-1/2)(i+1/2)}{i^2}$. Then,

$$\frac{3}{4k^2} \leq v \leq 1$$

Proof. We have that $v = \prod_{i=1}^{k} \left(1 - \frac{1}{4i^2}\right)$, which is a product of numbers less than 1 and justifies the upper bound. For the lower bound, we have:

$$\begin{aligned}
\frac{1}{v} &= \prod_{i=1}^{k} \frac{4i^2}{4i^2-1} \\
&= \frac{4}{3} \cdot \frac{16}{15} \cdot \ldots \cdot \frac{4k^2}{4k^2-1} \\
&\leq \frac{4}{3} \cdot \frac{16}{4} \cdot \ldots \cdot \frac{4k^2}{4(k-1)^2} \\
&= \frac{4k^2}{3}.
\end{aligned}$$

◁

B Proof of Lemma 16

Proof. We find the maximising value of C_w by expanding the expression for C_w and removing terms that do not depend on w:

$$\begin{aligned}
\arg\max_w C_w &= \arg\max_w \frac{\binom{k}{w}\binom{n-k}{\frac{1}{2}(n-k)}}{\binom{n}{\frac{1}{2}(n-k+2w)}} \cdot \frac{n^k \cdot (n-k)!!^2}{(n-k+2w)!! \cdot (n-2w+k)!!} \\
&= \arg\max_w \frac{\binom{k}{w}}{\binom{n}{\frac{1}{2}(n-k+2w)}} \cdot \frac{1}{(n-k+2w)!! \cdot (n-2w+k)!!} \\
&= \arg\max_w \frac{k! \cdot (\frac{n-k+2w}{2})! \cdot (n-\frac{n-k+2w}{2})!}{w! \cdot (k-w)! \cdot n! \cdot (n-k+2w)!! \cdot (n-2w+k)!!} \\
&= \arg\max_w \frac{(\frac{n-k+2w}{2})! \cdot (\frac{n+k-2w}{2})!}{w! \cdot (k-w)! \cdot (n-k+2w)!! \cdot (n-2w+k)!!} \\
&= \arg\max_w \frac{(\frac{n-k+2w}{2})! \cdot (\frac{n+k-2w}{2})!}{w! \cdot (k-w)! \cdot 2^n \cdot (\frac{n-k+2w}{2})! \cdot (\frac{n-2w+k}{2})!} \\
&= \arg\max_w \frac{1}{w! \cdot (k-w)!} = k/2
\end{aligned}$$

◀

C Proof of Corollary 3

In this section we justify Corollary 3. The proof will rely on two technical lemmas which are presented prior to the final proof of the corollary.

▶ **Lemma 21.** *Let $g = g(\epsilon)$ be defined over $[0, 0.4)$, where g is:*

$$\frac{(1.6+\epsilon)^{0.8+\epsilon/2} \cdot (0.4-\epsilon)^{0.2-\epsilon/2}}{2^{0.6+\epsilon}}$$

Then, for any positive ϵ in the domain of g, we have $g(\epsilon) > 4/5$.

Proof. It suffices to show: (a) that g is minimised over its domain at 0 and $g(0) = 4/5$ and (b) that $g' > 0$ for all positive ϵ. Computing the derivative, we have:

$$g'(\epsilon) = (1/5) \cdot \left(2^{-1.6-\epsilon} \cdot (2-5\epsilon)^{0.2-\epsilon/2} \cdot (8+5\epsilon)^{0.8+\epsilon/2} \cdot (\ln(1.6+\epsilon) - \ln(1.6-4\epsilon))\right)$$

The statement (a) is justified by checking that $4/5 = g(0) \leq \lim_{\epsilon\to 0.4} g(\epsilon)$ and checking that the derivative is zero only at $\epsilon = 0$. Statement (b) follows from observing that g' is a product of exponential terms that are always positive and the factor $\ln(1.6+\epsilon) - \ln(1.6-4\epsilon)$, which is also always positive for positive ϵ because the natural log is an increasing function. ◀

▶ **Lemma 22.** *Let f_n denote the function of k in the lower bound of Theorem 2, when n is sufficiently large and fixed. Similarly, let F_n be the function of k in the upper bound Theorem 1 when n is sufficiently large and fixed. Then, for any fixed $\epsilon > 0$, $F_n(0.6n) < f_n((0.6+\epsilon)n)$.*

Proof. Fix $\epsilon > 0$ and let δ be $g(\epsilon) - 4/5$. By Lemma 21, $\delta > 0$. The present lemma then follows immediately from the inequalities:

$$F_n(0.6n) = O(n^c) \cdot (4/5)^n \leq (4/5+\delta)^n \leq g^n(\epsilon) = f_n((0.6+\epsilon)n),$$

where the first equality is from Theorem 1, the first inequality is true for large enough n, the second inequality follows from Lemma 21, and the final equality is from the definition of f_n and Theorem 2. ◀

We now use Lemmas 21 and 22 to prove Corollary 3.

Proof. Fix arbitrary $\epsilon > 0$. In the game of Corollary 3, suppose that Player 1 chooses $\tilde{k}$, where

$$\tilde{k} > (0.6 + \epsilon) \cdot n.$$

Then, by Theorem 2 and Lemma 22, Player 2 will be able to choose a symmetric pair of perfectly $(\tilde{k} - 1)$-wise indistinguishable distributions where the $\tilde{k}$-wise reconstruction advantage is at least $f_n(\tilde{k}) > F_n(0.6n)$. Thus, the payoff of Player 1 will be strictly less than $1 - F_n(0.6n)$. This $\tilde{k}$ could not have been the optimal strategy for Player 1, since Player 1 can choose $0.6n$, where by Theorem 1, Player 2 achieves payoff at most $F_n(0.6n)$ and Player 1 gets payoff at least $1 - F_n(0.6n)$. Thus, the optimal strategy k^* cannot be chosen to be as large as $\tilde{k}$ and we have that

$$k^* - 0.6n \leq \epsilon n$$

We omit the proof of the complementary lower bound on k^* because it follows exactly the same structure as the upper bound. Together, these bounds imply that $|k^* - 0.6n| \leq \epsilon n$ for any arbitrarily small positive constant ϵ. ◀

D Proof of Corollary 4

Proof. Suppose that there exists an f that obeys the premises of the corollary, namely symmetry, no low-degree terms, and symmetry of Fourier coefficients. Define distributions μ, ν from f as follows (as in the method of [2]): $\mu(z) = 2 \cdot \max\{0, f(z)\}, \nu(z) = 2 \cdot \max\{0, -f(z)\}$. The total weight of each distribution is 1 because $\sum_z |f(z)| = 1$ and because by assumption $\hat{f}(\varnothing) = 0$, which implies that $\sum_z f(z) = 0$. Thus μ and ν are valid distributions. Next, observe that for every function $\chi_S := \prod_{i \in S} x_i$, the function χ_S has zero distinguishing advantage between μ and ν when $|S| \leq k - 1$:

$$\hat{f}(S) = 0 \implies \sum_z \chi_S(z) f(z) = 0 \implies \frac{1}{2}\left(\sum_z \chi_S(z)\mu(z) - \sum_z \chi_S(z)\nu(z)\right) = 0$$

$$\implies \mathbb{E}_{X\sim\mu}[\chi_S(X|_S)] - \mathbb{E}_{Y\sim\nu}[\chi_S(Y|_S)] = 0.$$

The set of all χ_S for $|S| \leq k - 1$ form a basis for all statistical tests on $k - 1$ bits; thus it follows immediately that μ and ν are perfectly $(k-1)$-wise indistinguishable. The symmetry of μ and ν follows straightforwardly from the symmetry of the Fourier coefficients of f.

We then have that for any $S \subseteq [n]$ where $|S| = k$,

$$\hat{f}(S) = \mathbb{E}[\chi_S \cdot f] = \frac{1}{2}\mathbb{E}[\chi_S \cdot (\mu - \nu)] = \mathbb{E}_{X\sim\mu}[\chi_S(X|_S)] - \mathbb{E}_{Y\sim\nu}[\chi_S(Y|_S)]$$
$$\leq O(n^c) \cdot \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}$$

The final inequality follows from Theorem 1 and fact that μ and ν obey the premises of that theorem, the implication that the statistical distance between any k-wise marginals of μ and ν is at most $O(n^c) \cdot \frac{(n-k)^{\frac{n-k}{2}} \cdot (n+k)^{\frac{n+k}{2}}}{2^k \cdot n^n}$, the fact that χ_S is a statistical test on k bits, and the fact that $\mathbb{E}_{X\sim\mu}[\chi_S(X|_S)] - \mathbb{E}_{Y\sim\nu}[\chi_S(Y|_S)]$ is precisely the reconstruction advantage of the statistical test χ_S. ◀

Analyzing XOR-Forrelation Through Stochastic Calculus

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

Abstract

In this note we present a simplified analysis of the quantum and classical complexity of the k-XOR Forrelation problem (introduced in the paper of Girish, Raz and Zhan [7]) by a stochastic interpretation of the Forrelation distribution.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory

Keywords and phrases quantum complexity, Brownian motion

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.60

Related Version *Full Version*: https://arxiv.org/abs/2109.02732

Funding Supported by NSF grant CCF-1717606. This material is based upon work supported by the National Science Foundation under grant numbers listed above. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation (NSF).

Acknowledgements I would like to thank Ryan O'Donnell for many helpful comments on this paper.

1 Introduction

The Forrelation problem [1] and variants of it have been useful in producing problems that are efficiently solvable by quantum protocols but are hard for classical protocols, in various different models. A recent line of work analyzing the Forrelation distribution builds on the polarizing random walk framework introduced by Chattopadhyay, Hatami, Hosseini and Lovett [4]. This framework views the Forrelation distribution as being generated by a random walk in $\mathbb{R}^N$, producing a particular Gaussian distribution, and then rounded to the Boolean cube $\{-1, 1\}^N$. This approach lead to breakthroughs as Raz and Tal's result on the oracle separation of BQP and PH [9] and Bansal and Sinha's proof that k-Forrelation exhibits an optimal separation between quantum and classical query complexity [2][1].

The recent work of Girish, Raz and Zhan [7] analyzes the XOR of k copies of the Forrelation function, and shows that the resulting problem is such that classical protocols of quasipolynomial size can only achieve quasipolynomially small advantage over random guessing, while there exist quantum protocols with complexity polylog(N). They show this for quantum simultaneous-message communication protocols vs. classical randomized communication protocols, as well as for quantum query complexity vs. classical query complexity.

Stochastic calculus viewpoint

The approach here generalizes [13] (indeed, the $k = 1$ case is identical). There are two main points where the stochastic approach simplifies the argument in [7].

[1] The proof is phrased in terms of Gaussian interpolation, which is a different viewpoint on the stochastic approach.

First, the Forrelation distribution, prior to rounding, is a truncated multivariate Gaussian. A multivariate N-dimensional Gaussian can also be realized as an N-dimensional Brownian motion, stopped at some constant time. Using a continuous-time random walk allows us to apply stochastic calculus techniques to bound how well f distinguishes the two distributions directly using the $2k^{\text{th}}$ order derivatives of f, without the need for additional intermediate bounds. Furthermore, the Brownian motion approach allows for an induction on k, eliminating the need for complex dimension-dependent bounds.

Second, viewing the Gaussian as a Brownian motion also allows us to use a stopping time to encode the truncation. This allows us to directly encode the boundedness of the distribution in the random variable. This eliminates the extra step to truncate the Gaussian and bound the closeness in expectation between the truncated and non-truncated Gaussians.

Connections and future work

Conceptually, viewing a truncated Gaussian as a stopped Brownian motion enforces a pathwise view of the random variable, i.e. sampling from the distribution means sampling a path of a random walk. This makes calculations on the distributions easier, for instance because the paths naturally split into "paths which always remain within the region" and "paths which end by hitting the boundary". This technique may also be interesting for other applications using truncated Gaussians (or analogously, replacing a truncated exponential distribution by a stopped geometric Brownian motion.) The stochastic calculus view of Gaussians has also been useful for other Boolean analysis results, for instance in the proof of Bobkov's Two Point Inequality by Barthe and Maurey [3]. Ideas related to the pathwise view of random variables also appear in the recent paper of Eldan and Gross [6], which expresses the variance and influence of a Boolean function in terms of its action on a certain Brownian motion.

2 Preliminaries

We state the main stochastic calculus result we will need in the proof. This is Dynkin's formula [8, Theorem 7.4.1] specialized to our scenario of the Brownian motion having mean 0 and constant covariance.

▶ **Theorem 1.** *Let $\mathbf{X}$ be an n-dimensional Brownian motion with mean 0 and covariance Σ, let τ be a bounded stopping time, and let $f : \mathbb{R}^N \to \mathbb{R}$ be a twice continuously differentiable function. We use $\mathrm{H}f$ to denote the Hessian of f, the $N \times N$ matrix of second order partial derivatives. The following holds:*

$$\mathbf{E}[f(\mathbf{X}_\tau)] = f(0) + \mathbf{E}\left[\int_0^\tau \frac{1}{2}\langle \Sigma, \mathrm{H}f(\mathbf{X}_s)\rangle \, ds\right].$$

We also need the following formula regarding random restrictions, which is essentially Lemma 1 of [13]. A similar idea appears in the proof of Lemma 5.1 of [7], and previously in [5, Claim A.5].

▶ **Lemma 2.** *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a multilinear polynomial. For any $x \in [-1/2, 1/2]^N$, there exists a distribution $\mathcal{R}_x$ over restrictions $\rho \in \{-1, 1, *\}^N$, such that for any $S \subseteq [N]$,*

$$\partial_S f(x) = 2^{|S|} \mathop{\mathbf{E}}_{\rho \sim \mathcal{R}_x}[\partial_S f_\rho(0)].$$

Here we write $\partial_S = \prod_{i \in S} \frac{\partial}{\partial_i}$ *for the partial derivatives over the coordinates in* S*. We further define the* Fourier coefficient $\widehat{f}(S) := \partial_S f(0)$*. Note that this coincides with the usual decomposition* $f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \prod_{i \in S} x_i$*.*

3 A bound on a product of Brownian motions

▶ **Definition 3.** *Let* $k \in \mathbb{N}_+$*, and let* $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(k)}$ *be identical independent* N*-dimensional Brownian motions with mean 0 and covariance matrix* Σ*, and let* $\tau_1, \ldots, \tau_k$ *be stopping times.*

We consider distributions on $\mathbb{R}^{kN} \cong (\mathbb{R}^N)^k$*, which we take to be* k *copies of* $\mathbb{R}^N$*, indexed by coordinates* $1, \ldots, k$*. Let* $S \subseteq [k]$*. Define the random variable* $\mathbf{X}_\tau^S$ *to be* $\mathbf{X}_{\tau_i}^{(i)}$ *in the* i^{th} *coordinate if* $i \in S$*, and* 0 *in the* i^{th} *coordinate if* $i \notin S$*. We set* $\mathbf{D}_S$ *to be the distribution of* $\mathbf{X}_\tau^S$*.*

We write $\mathbf{S} \sim [k]$ *to denote drawing* $\mathbf{S} \subseteq [k]$ *uniformly. We now define the distribution* $\mathbf{D}_{\text{odd},\,k}$ *to be the distribution of* $\mathbf{D}_\mathbf{S}$ *conditioned on* $|\mathbf{S}|$ *being odd. Similarly, we define* $\mathbf{D}_{\text{even},\,k}$ *to be* $\mathbf{D}_\mathbf{S}$ *conditioned on* $|\mathbf{S}|$ *being even. When* $k = 1$*, we define* $\mathbf{D}_1 = \mathbf{X}_{\tau_1}^{(1)} = \mathbf{D}_{\text{odd},\,1}$*.*

For a multilinear function $f : \mathbb{R}^{kN} \to \mathbb{R}$*, we note the identity*

$$\mathbf{E}[f(\mathbf{D}_{\text{even},\,k})] - \mathbf{E}[f(\mathbf{D}_{\text{odd},\,k})] = 2 \mathop{\mathbf{E}}_{\mathbf{S} \sim [k]} \Big[(-1)^{|\mathbf{S}|} f(\mathbf{D}_\mathbf{S}) \Big]. \tag{1}$$

The following bounds how well a Boolean function with bounded level-$2k$ Fourier weight can distinguish $\mathbf{D}_{\text{even},\,k}$ and $\mathbf{D}_{\text{odd},\,k}$. This is essentially Theorem 3.1 of [7].

▶ **Theorem 4.** *Let* $k \in \mathbb{N}_+$*, let* $f : \{-1,1\}^{kN} \to \{-1,1\}$ *be a Boolean function, and let* $L > 0$ *be such that for any restriction* ρ*,*

$$\sum_{\substack{S \subseteq [kN] \\ |S| = 2k}} |\widehat{f}_\rho(S)| \le L.$$

Let $\gamma > 0$ *and let* $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(k)}$ *be identical independent* N*-dimensional Brownian motions with mean 0 and covariance matrix* Σ*. Further assume that* $|\Sigma_{ij}| \le \gamma$ *for* $i \ne j$*.*

Let $\varepsilon > 0$ *and define the (bounded) stopping times for each* $i \in [k]$*,*

$$\tau_i := \min\{\varepsilon,\ \textit{first time that } \mathbf{X}^{(i)} \textit{ exits } [-1/2, 1/2]^N\}.$$

Then, identifying f *with its multilinear expansion, we have*

$$|\mathop{\mathbf{E}}_{\mathbf{S} \sim [k]} \Big[(-1)^{|\mathbf{S}|} f(\mathbf{D}_\mathbf{S}) \Big]| \le (\varepsilon\gamma)^k L.$$

Proof. We first prove by induction on k that for any multilinear function f,

$$\mathop{\mathbf{E}}_{\mathbf{S} \sim [k]} \big[(-1)^{|\mathbf{S}|} f(\mathbf{D}_\mathbf{S}) \big] = \mathbf{E}\left[\int_0^{\tau_1} \cdots \int_0^{\tau_k} \frac{(-1)^{k-1}}{2^{2k-1}} \Big\langle (I_k \otimes \Sigma)^{\otimes k},\ \mathrm{H}^{\otimes k} f(\mathbf{X}_{t_1}^{(1)}, \ldots, \mathbf{X}_{t_k}^{(k)}) \Big\rangle \, dt_1 \ldots dt_k \right], \tag{2}$$

where $\mathrm{H}^{\otimes k} f$ denotes the $(kN)^k$-dimensional matrix of all the $2k^{\text{th}}$ order derivatives of f, I_k is the k-dimensional identity matrix, and $\otimes$ denotes the Kronecker product.

The base case $k = 1$ is simply a direct application of Dynkin's formula (Theorem 1):

$$\mathbf{E}[f(\mathbf{X}_{\tau_1}^{(1)})] - f(0) = \mathbf{E}\left[\int_0^{\tau_1} \frac{1}{2} \Big\langle \Sigma,\ \mathrm{H} f(\mathbf{X}_{t_1}^{(1)}) \Big\rangle \, dt_1 \right].$$

For the induction step, we condition on the last coordinate to observe that

$$\mathop{\mathbf{E}}_{\mathbf{S}\sim[k]}\Big[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}})\Big] = \frac{1}{2}\mathop{\mathbf{E}}_{\mathbf{S}\sim[k-1]}\Big[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}},0)\Big] - \frac{1}{2}\mathop{\mathbf{E}}_{\mathbf{S}\sim[k-1]}\Big[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}},\mathbf{X}^{(k)}_{\tau_k})\Big] \tag{3}$$

Now let $\mathbf{S} \sim [k-1]$. We will proceed by applying Dynkin's formula to $g(x) = \mathbf{E}_{\mathbf{S},\,\mathbf{D}_{\mathbf{S}}}[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}},x)]$. Since f is a multilinear function, partial derivatives of g commute with the expectation; in particular $\partial_i g(x) = \mathbf{E}[(-1)^{|\mathbf{S}|}\partial_{i+(k-1)N}f(\mathbf{D}_{\mathbf{S}},x)]$, and so, with $e_k \in \mathbb{R}^k$ denoting the indicator of the k^{th} coordinate,

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{S},\,\mathbf{D}_{\mathbf{S}},\,\mathbf{X}^{(k)},\,\tau_k}&[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}},\mathbf{X}^{(k)}_{\tau_k})] - \mathop{\mathbf{E}}_{\mathbf{S},\,\mathbf{D}_{\mathbf{S}}}[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}},0)]\\ &= \mathop{\mathbf{E}}_{\mathbf{X}^{(k)},\,\tau_k}\left[\frac{1}{2}\int_0^{\tau_k}\left\langle e_k e_k^T\otimes\Sigma, \mathop{\mathbf{E}}_{\mathbf{S},\,\mathbf{D}_{\mathbf{S}}}\Big[(-1)^{|\mathbf{S}|}\,\mathrm{H}f(\mathbf{D}_{\mathbf{S}},\mathbf{X}^{(k)}_{t_k})\Big]\right\rangle dt_k\right]\end{aligned} \tag{4}$$

Finally, we apply the induction hypothesis to find, for $(k-1)N < i,j \le kN$,

$$\begin{aligned}&\mathop{\mathbf{E}}_{\mathbf{S}\sim[k-1]}\Big[(-1)^{|\mathbf{S}|}\partial_{i,j}f(\mathbf{D}_{\mathbf{S}},\mathbf{X}^{(k)}_{t_k})\Big]\\ &= \mathbf{E}\left[\int_0^{\tau_1}\!\!\cdots\int_0^{\tau_{k-1}}\frac{(-1)^{k-2}}{2^{2k-3}}\Big\langle(I_{k-1}\otimes\Sigma)^{\otimes(k-1)},\ \mathrm{H}^{\otimes(k-1)}_{\mathbf{X}^{(1)},\dots,\mathbf{X}^{(k-1)}}\partial_{i,j}f(\mathbf{X}^{(1)}_{t_1},\dots,\mathbf{X}^{(k)}_{t_k})\Big\rangle\,dt_1\dots dt_{k-1}\right].\end{aligned}$$

Combining with Equations (3) and (4) and using bilinearity of the inner product, we conclude

$$\begin{aligned}&\mathop{\mathbf{E}}_{\mathbf{S}\sim[k]}\Big[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}})\Big]\\ &\quad= -\frac{1}{2}\mathop{\mathbf{E}}_{\mathbf{X}^{(k)},\,\tau_k}\left[\frac{1}{2}\int_0^{\tau_k}\left\langle e_k e_k^T\otimes\Sigma, \mathop{\mathbf{E}}_{\mathbf{S},\,\mathbf{D}_{\mathbf{S}}}\Big[(-1)^{|\mathbf{S}|}\,\mathrm{H}f(\mathbf{D}_{\mathbf{S}},\mathbf{X}^{(k)}_{t_k})\Big]\right\rangle dt_k\right]\\ &\quad= \mathbf{E}\Bigg[\int_0^{\tau_1}\!\!\cdots\int_0^{\tau_k}\frac{(-1)^{k-1}}{2^{2k-1}}\Big\langle e_k e_k^T\otimes\Sigma, \mathrm{H}_{\mathbf{X}^{(k)}}\Big\langle(I_{k-1}\otimes\Sigma)^{\otimes(k-1)},\\ &\qquad\qquad \mathrm{H}^{\otimes(k-1)}_{\mathbf{X}^{(1)},\dots,\mathbf{X}^{(k-1)}}f(\mathbf{X}^{(1)}_{t_1},\dots,\mathbf{X}^{(k)}_{t_k})\Big\rangle\Big\rangle\,dt_1\dots dt_k\Bigg]\\ &\quad= \mathbf{E}\left[\int_0^{\tau_1}\!\!\cdots\int_0^{\tau_k}\frac{(-1)^{k-1}}{2^{2k-1}}\Big\langle(I_k\otimes\Sigma)^{\otimes k},\ \mathrm{H}^{\otimes k}f(\mathbf{X}^{(1)}_{t_1},\dots,\mathbf{X}^{(k)}_{t_k})\Big\rangle\,dt_1\dots dt_k\right].\end{aligned}$$

Having completed the proof of Equation (2), we now use it to prove the theorem

$$\begin{aligned}&\mathop{\mathbf{E}}_{\mathbf{S}\sim[k]}\Big[(-1)^{|\mathbf{S}|}f(\mathbf{D}_{\mathbf{S}})\Big]\\ &\quad\le \varepsilon^k\,\mathbf{E}\left[\sup_{\substack{t_1\in[0,\tau_1]\\ \cdots\\ t_k\in[0,\tau_k]}}\Big|\frac{1}{2^{2k}}\Big\langle(I_k\otimes\Sigma)^{\otimes k},\ \mathrm{H}^{\otimes k}f(\mathbf{X}^{(1)}_{t_1},\dots,\mathbf{X}^{(k)}_{t_k})\Big\rangle\Big|\right] && (\tau_1,\dots,\tau_k\le\varepsilon)\\ &\quad\le \frac{(\varepsilon\gamma)^k}{2^{2k}}\sup_{(x_1,\dots,x_k)\in[-1/2,1/2]^{kN}}\sum_{\substack{S\subseteq[kN]\\|S|=2k}}|\partial_S f(x_1,\dots,x_k)| && (|\Sigma_{ij}|\le\gamma \text{ for } i\neq j,\ \partial_{ii}f=0)\\ &\quad\le (\varepsilon\gamma)^k\sup_{(x_1,\dots,x_k)\in[-1/2,1/2]^{kN}}\sum_{\substack{S\subseteq[kN]\\|S|=2k}}\Big|\mathop{\mathbf{E}}_{\rho\sim\mathcal{R}_{x_1,\dots,x_k}}[\partial_S f_\rho(0,\dots,0)]\Big| && (\text{Lemma 2})\\ &\quad\le (\varepsilon\gamma)^k\sup_{(x_1,\dots,x_k)\in[-1/2,1/2]^{kN}}\mathop{\mathbf{E}}_{\rho\sim\mathcal{R}_{x_1,\dots,x_k}}\left[\sum_{\substack{S\subseteq[kN]\\|S|=2k}}|\widehat{f_\rho}(S)|\right]\\ &\quad\le (\varepsilon\gamma)^k L.\end{aligned}$$

◄

4 Application to complexity of k-XOR Forrelation

We now apply the bound from the previous section to prove the main theorem from [7, Theorem 3.1], from which they derive separations in quantum versus classical query complexity, communication complexity and circuit complexity (we refer to [7] for the exact details about the definitions of the complexity classes and the full proof).

We briefly sketch how the proof in [7] proceeds. For the lower bounds on the classical complexity classes, it suffices to exhibit two distributions that are hard for functions in the complexity class to distinguish. These will be derived from $\mathbf{D}_{\text{odd},\,k}$ and $\mathbf{D}_{\text{even},\,k}$, with Σ and ε chosen appropriately ([7] uses the truncated Gaussian instead of the stopping time here). [7] observes that these classical complexity classes are closed under restrictions. Then, Theorem 4 and Equation (1) combined with bounds on the level-k Fourier weights proven in [11] and [7] shows the classical lower bounds.

For the quantum upper bound, we need to show that a quantum query algorithm (or communication protocol respectively) can distinguish $\mathbf{D}_{\text{odd},\,k}$ and $\mathbf{D}_{\text{even},\,k}$ with high probability. We will show that the concentration results proven in [7] hold in our context as well.

We now set values for ε and k. We take $\varepsilon = 1/(28k^2 \ln N)$, and k small enough that $\varepsilon^2 N \leq \text{poly}(N)$ (e.g. $k \leq O(N^{1/5})$ suffices), and set

$$\Sigma := \begin{pmatrix} I_n & H_n \\ H_n & I_n \end{pmatrix},$$

where $N = 2n$, n is a power of 2 and H_n is the normalized Hadamard matrix, so $\gamma = \frac{1}{\sqrt{n}}$. Applying Theorem 4, the overall upper bound is $L_{2k} \cdot \text{polylog}(N)/N^k$, where L_{2k} is the bound on the Fourier weight at level $2k$ for the family of functions in the complexity class.

The quantum algorithm/communication protocol is based on the k-XOR Forrelation problem, which we define here: Let $\phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as $\phi(x, y) := \frac{1}{n}\langle x, H_n y\rangle$. The Forrelation decision problem is a partial function defined by

$$F(x, y) = \begin{cases} -1 & \text{if } \phi(x, y) \geq \varepsilon/2, \\ 1 & \text{if } \phi(x, y) \leq \varepsilon/4. \end{cases}$$

The k-XOR Forrelation $F^{(k)} : \{-1, 1\}^{kN} \to \{-1, 1\}$ is defined by $F^{(k)}(z_1, \ldots, z_k) = \prod_{i=1}^{k} F(z_i)$.

Since $\mathbf{D}_{\text{odd},\,k}$ and $\mathbf{D}_{\text{even},\,k}$ take values in $[-1/2, 1/2]^{kN}$ but $F^{(k)}$ is defined on $\{-1, 1\}^{kN}$, we round them to distributions $\widetilde{\mathbf{D}}_{\text{odd},\,k}$ and $\widetilde{\mathbf{D}}_{\text{even},\,k}$ on $\{-1, 1\}^{kN}$. A draw of $\widetilde{\mathbf{z}} \sim \widetilde{\mathbf{D}}_{\text{odd},\,k}$ is defined as follows:

nosep Sample $\mathbf{z} \sim \mathbf{D}_{\text{odd},\,k}$.

nosep For each coordinate $i \in [N]$, independently set $\widetilde{\mathbf{z}}_i = 1$ with probability $\frac{1+\mathbf{z}_i}{2}$ and -1 with probability $\frac{1-\mathbf{z}_i}{2}$. We denote $\widetilde{\mathbf{z}} \sim \mathbf{z}$ for this step. Now $\widetilde{\mathbf{z}}$ is sampled from $\widetilde{\mathbf{D}}_{\text{odd},\,k}$.

$\widetilde{\mathbf{D}}_{\text{even},\,k}$ is defined analogously. Note that for a multilinear polynomial $f : \mathbb{R}^N \to \mathbb{R}$, $\mathbf{E}[f(\mathbf{D}_{\text{odd},\,k})] = \mathbf{E}[f(\widetilde{\mathbf{D}}_{\text{odd},\,k})]$ (analogously for $\widetilde{\mathbf{D}}_{\text{even},\,k}$).

Girish, Raz and Zhan showed the following about the rounding process:

▶ **Proposition 5** (Claim A.2 [7]). *Let $z \in [-1/2, 1/2]$, and let $\widetilde{\mathbf{z}} \sim z$ as in step 2 above. Then,*

$$\mathbf{P}[|\phi(\widetilde{\mathbf{z}}) - \phi(z)| > \varepsilon/4] \leq \exp(-\Omega(N^{1/4})).$$

Finally, we show a concentration result analogous to Lemma 2.11 of [7] which shows that $F^{(k)}$ decides correctly on $\widetilde{\mathbf{D}}_{\text{odd},\,k}$ and $\widetilde{\mathbf{D}}_{\text{even},\,k}$ with high probability. This is then sufficient to deduce the applications described in [7].

First, we prove a concentration bound for $\phi(\mathbf{D}_1)$, i.e. $(\mathbf{x}, \mathbf{y})$ are generated by a single N-dimensional stopped Brownian motion with covariance Σ.

▶ **Lemma 6.** *In the above context, the following holds:*

$$\mathop{\mathbf{P}}_{(\mathbf{x},\mathbf{y})\sim\mathbf{D}_1}[\phi(\mathbf{x}, \mathbf{y}) \geq 3\varepsilon/4] \geq 1 - O(1/N^{6k^2}). \tag{5}$$

Proof. Notice that an alternate way to sample $(\mathbf{x}, \mathbf{y}) \sim \mathbf{D}_1$ is to let $\mathbf{X}_t$ be a n-dimensional Brownian motion with covariance I_n stopped at the stopping time

$$\tau := \min\{\varepsilon, \text{ first time that } \mathbf{X}_t \text{ or } H_n\mathbf{X}_t \text{ exits } [-1/2, 1/2]^n\},$$

and let $(\mathbf{x}, \mathbf{y}) = (\mathbf{X}_\tau, H_n\mathbf{X}_\tau)$. Then, $\phi(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\|\mathbf{X}_\tau\|_2^2$. In order to prove the desired bound, we first prove that with high probability $\tau = \varepsilon$, i.e. the path of the Brownian motion did not exit $[-1/2, 1/2]^N$ before time ε. We then show that $\frac{1}{n}\|\mathbf{X}_\varepsilon\|_2^2 \geq 3\varepsilon/4$ with high probability, and conclude using a union bound. We can union bound over the N coordinates,

$$\mathbf{Pr}[\tau < \varepsilon] \leq N \cdot \mathbf{P}[\text{1st coordinate of } X_t \text{ exits } [-1/2, 1/2] \text{ earlier than } \varepsilon/2].$$

Since each coordinate of $\mathbf{X}_t$ is a standard 1D Brownian motion $\mathbf{B}_t$, we can apply Doob's submartingale inequality (e.g. [10, Proposition II.1.8]) to obtain

$$\mathbf{Pr}\left[\sup_{0\leq t\leq\varepsilon/2} |\mathbf{B}_t| \geq \frac{1}{2}\right] \leq 2e^{-1/4\varepsilon} = 2e^{-7k^2\ln N} = \frac{2}{N^{7k^2}}.$$

Therefore,

$$\mathbf{Pr}[\tau < \varepsilon] \leq 2/N^{7k^2-1}. \tag{6}$$

Next, we consider $\frac{1}{n}\|\mathbf{X}_\varepsilon\|_2^2$. Note that this is simply the average of the squares of n iid Gaussians $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with mean 0 and variance ε. Using [12, Example 2.11], we have the tail bound

$$\mathbf{Pr}\left[|\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^2 - \varepsilon| \geq \frac{\varepsilon}{4}\right] \leq \exp(-\Omega(N)). \tag{7}$$

Taking a union bound over Equations (6) and (7), we have $\mathbf{P}_{(\mathbf{x},\mathbf{y})\sim\mathbf{D}_1}[\phi(\mathbf{x}, \mathbf{y}) \leq 3\varepsilon/4] \leq O(1/N^{6k^2})$. ◀

▶ **Proposition 7.** *The following hold:*

$$\mathop{\mathbf{P}}_{\widetilde{\mathbf{z}}\sim\widetilde{\mathbf{D}}_{\text{even},\,k}}[F^{(k)}(\widetilde{\mathbf{z}}) = 1] \geq 1 - O\Big(\frac{k}{N^{6k^2}}\Big) \qquad \textit{and} \qquad \mathop{\mathbf{P}}_{\widetilde{\mathbf{z}}\sim\widetilde{\mathbf{D}}_{\text{odd},\,k}}[F^{(k)}(\widetilde{\mathbf{z}}) = -1] \geq 1 - O\Big(\frac{k}{N^{6k^2}}\Big).$$

Proof. We first show F decides correctly on the coordinates with $\mathbf{U}_N$ with high probability:

$$\mathop{\mathbf{P}}_{(\mathbf{x},\mathbf{y})\sim\mathbf{U}_N}[\phi(\mathbf{x}, \mathbf{y}) \leq \varepsilon/4] \geq 1 - \exp(-\Omega(N\varepsilon^2)). \tag{8}$$

To see this, note that $\mathbf{x}$ and $\mathbf{y}$ are independent, so $\mathbf{x}$ and $H_n\mathbf{y}$ are independent. Hence $\phi(\mathbf{x}, \mathbf{y})$ is simply the average of random signs, and the bound holds by Hoeffding's inequality.

Finally, to prove the proposition it suffices to prove that for any fixed $S \subseteq [k]$, $\mathbf{P}_{\mathbf{z}\sim\mathbf{D}_S, \widetilde{\mathbf{z}}\sim\mathbf{z}}[F^{(k)}(\widetilde{\mathbf{z}}) \neq (-1)^{|S|}] \leq O(k/N^{6k^2})$. If $i \in S$, then $\widetilde{\mathbf{z}}_i$ is distributed as $\widetilde{\mathbf{D}}_1$, so Lemma 6 combined with Proposition 5 implies $\mathbf{P}[F(\widetilde{\mathbf{z}}_i) = 1] \leq O(1/N^{6k^2})$. Meanwhile if $i \notin S$, then $\widetilde{\mathbf{z}}_i$ is distributed as $\mathbf{U}_N$, so Equation (8) implies $\mathbf{P}[F(\widetilde{\mathbf{z}}_i) = -1] \leq \exp(-\Omega(N\varepsilon^2))$. With k and therefore ε taken sufficiently small, a union bound over the k coordinates completes the proof. ◀

References

1 Scott Aaronson and Andris Ambainis. Forrelation: a problem that optimally separates quantum from classical computing. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 307–316, 2015.

2 Nikhil Bansal and Makrand Sinha. k-Forrelation optimally separates quantum and classical query complexity. Technical Report 2008.07003, arXiv, 2020.

3 F. Barthe and B. Maurey. Some remarks on isoperimetry of Gaussian type. *Ann. Inst. H. Poincaré Probab. Statist.*, 36(4):419–434, 2000. `doi:10.1016/S0246-0203(00)00131-X`.

4 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. *Theory Comput.*, 15:Paper No. 10, 26, 2019. `doi:10.4086/toc.2019.v015a010`.

5 Eshan Chattopadhyay, Pooya Hatami, Shachar Lovett, and Avishay Tal. Pseudorandom generators from the second Fourier level and applications to AC0 with parity gates. In *Proceedings of the 10th Annual Innovations in Theoretical Computer Science Conference*, pages 22:1–22:15, 2018.

6 Ronen Eldan and Renan Gross. Concentration on the Boolean hypercube via pathwise stochastic analysis. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing*, pages 208–221. ACM, New York, 2020.

7 Uma Girish, Ran Raz, and Wei Zhan. Lower bounds for XOR of forrelations. Technical report, arXiv, 2020. `arXiv:2007.03631`.

8 Bernt Øksendal. *Stochastic differential equations*. Universitext. Springer-Verlag, Berlin, sixth edition, 2003. An introduction with applications. `doi:10.1007/978-3-642-14394-6`.

9 Ran Raz and Avishay Tal. Oracle separation of BQP and PH. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, pages 13–23. ACM, New York, 2019. `doi:10.1145/3313276.3316315`.

10 Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, third edition, 1999. `doi:10.1007/978-3-662-06400-9`.

11 Avishay Tal. Towards optimal separations between quantum and randomized query complexities. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*, pages 228–239. IEEE Computer Soc., Los Alamitos, CA, 2020. `doi:10.1109/FOCS46700.2020.00030`.

12 Martin J. Wainwright. *High-dimensional statistics*, volume 48 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2019. A non-asymptotic viewpoint. `doi:10.1017/9781108627771`.

13 Xinyu Wu. A stochastic calculus approach to the oracle separation of BQP and PH. Technical report, arXiv, 2020. `arXiv:2007.02431`.